thermopack

v0.1

Generated by Doxygen 1.9.8

Chapter 1

Todo List

Subprogram critical::calccritical (t, p, z, phase, ierr, tol)

Add v=sqrt(z) to paramater vector

Module extcsp

Need trace-component functionality.

Module optimizers

Add special treatment for n=2 systems. Analytical modification of eigenvalues.

Module ps_solver

Need trace-component functionallity.

Subprogram spinodal::rho_of_meta_extremum (t, x, phase, rho_init_in)

: May need more sophisticated method of choosing initial liquid rho.

: May need more robust handling of overshoots. Now we just "hope".

: May need to check that we have converged to the *correct* extremum.

Module stability

Add termination when approaching trivial solution.

Module state_functions

Need trace-component functionallity.

Module sv_solver

Need trace-component functionallity.

Consider merging with UV-flash

Subprogram sv_solver::twophasesvsinglecomp (t, p, z, beta, betal, x, y, sspec, vspec, phase, ierr)

Need handling of solutions close to critical point

Module tp_solver

Need trace-component functionallity.

Add DEM of order 2 for accelration of multi-phase Rachford-Rice

Module uv_solver

Need trace-component functionallity.

Subprogram uv_solver::twophaseuvsinglecomp (t, p, z, beta, betal, x, y, uspec, vspec, phase, ierr)

Need handling of solutions close to critical point

Chapter 2

Modules Index

2.1 Modules List

Here is a list of all documented modules with brief descriptions:

Chapter 3

Data Type Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Data Type Index

4.1 Data Types List

Here are the data types with brief descriptions:

Chapter 5

Module Documentation

5.1 apparent_compostion Module Reference

Code for handling apparent composition approach See XMHX.

Data Types

- type apparent_container

Functions/Subroutines

- type(apparent_container) function, public **apparent_constructor** (nc, nce)
- subroutine **dealloc** (apparent)
- subroutine **set_v_stoich_ij** (apparent, i, j, v)
- subroutine **update_v_sum** (apparent)
- subroutine **apparent_to_real_mole_numbers** (apparent, n, ne)

  Map apparent mole numbers to real mole numbers used by model.
- subroutine **real_to_apparent_differentials** (apparent, fe_n, fe_tn, fe_vn, fe_nn, f_n, f_tn, f_vn, f_nn)

  Map from real mole number differentials to apparent mole number differentials.
- subroutine **real_to_apparent_diff** (apparent, fe_n, f_n)

  Map from real mole number differential to apparent mole number differentials.
- subroutine tp_lnfug_apparent (apparent, nc, ne, n, p, lnfug_real, lnfug, dlnfugdt_real, dlnfugdp_real, dlnfugdn_real, dlnfugdt, dlnfugdp, dlnfugdn)

  Convert logarithmic fugacity coefficient from real to apparent composition.
- subroutine getmodfugacity (apparent, t, p, x, lnfug, sumne)

  Back calculate logaritm of fugacity coefficient, by removing dependency of log(x) and log(xe)

5.1.1 Detailed Description

Code for handling apparent composition approach See XMHX.

Author

Morten Hammer, 2020

5.1.2 Function/Subroutine Documentation

5.1.2.1 getmodfugacity()

```
subroutine apparent_compostion::getmodfugacity (
             class(apparent_container), intent(in) apparent,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(apparent%nc), intent(in) x,
             real, dimension(apparent%nc), intent(inout) lnfug,
             real, intent(out) sumne )
```

Back calculate logaritm of fugacity coefficient, by removing dependency of log(x) and log(xe)

Author

MH, 2017-05

Parameters

in	*t*	Temperature (K)
in	*p*	Pressure (Pa)
in	*x*	Phase compozition

5.1.2.2 tp_lnfug_apparent()

```
subroutine apparent_compostion::tp_lnfug_apparent (
             class(apparent_container), intent(in) apparent,
             integer, intent(in) nc,
             real, dimension(apparent%nce), intent(in) ne,
             real, dimension(nc), intent(in) n,
             real, intent(in) p,
             real, dimension(apparent%nce), intent(in) lnfug_real,
             real, dimension(nc), intent(out) lnfug,
             real, dimension(apparent%nce), intent(in), optional dlnfugdt_real,
             real, dimension(apparent%nce), intent(in), optional dlnfugdp_real,
             real, dimension(apparent%nce,apparent%nce), intent(in), optional dlnfugdn_real,
             real, dimension(nc), intent(out), optional dlnfugdt,
             real, dimension(nc), intent(out), optional dlnfugdp,
             real, dimension(nc,nc), intent(out), optional dlnfugdn )
```

Convert logarithmic fugacity coefficient from real to apparent composition.

Author

Morten Hammer, 2017-03

Parameters

in	*n*	Apparent mole numbers [mols]
in	*p*	Pressure [Pa]
in	*ne*	Real mole numbers [mols]
~~in~~	~~*lnfug_real*~~	~~Log of real fugacities~~
out	*lnfug*	Log of apparent fugacity

5.2 association_var Module Reference

This module defines a type for association variables in the ThermoPack library.

Data Types

- type association
- type association_state

  Current state for eos evaluation.

Functions/Subroutines

- subroutine **dealloc** (assoc)
- subroutine **init_assoc_state** (assoc_p, nc, t, v, n)
- subroutine **init_assoc_state_fmt** (assoc_p, nc, t, n_fmt, m)
- subroutine **dealloc_assoc_state** (assoc_p)
- subroutine **print** (assoc)

5.2.1 Detailed Description

This module defines a type for association variables in the ThermoPack library.

5.3 assocschemeutils Module Reference

Functionality for working with association schemes. The work mostly consists of keeping track of indices.

Functions/Subroutines

- subroutine **associndices_bookkeeping** (assoc, nc, saft_model, assocschemes_db)
- integer function site_to_compidx (assoc, k)

  Gives the component number ic to which association site number k belongs.
- subroutine compidx_to_sites (assoc, ic, k_first, k_last)

  Gives the first and last association sites, k_first and k_last, for comp i.
- real function **applycombiningrule** (ruleidx, val1, val2)

  Implements the combining rules for eps and beta seen in CPA models. This utility is also used for PC-SAFT.
- logical function site_interaction_internal (site1, site2, assoc_scheme)

  Returns true if there is nonzero interaction between site1 and site2, according to association scheme assoc_scheme.
- logical function cross_site_interaction (site1, site2, assoc_scheme_i, assoc_scheme_ii)

  Given two associating components with different association schemes, one sometimes wants to model only the realistic site–site combinations on the different molecules, e.g. only the positively charged sites on ethanol should interact only with the negatively charged sites on H2O, and vice versa for negatively charged sites on ethanol and positively charged sites on H2O. This is the function which specifies this. It returns true if there is nonzero interaction between site1 on a molecule using scheme I and site2 on a molecule with scheme II.
- integer function **polarity** (site, scheme)

  Retrieve polarity of site given association scheme. If an association scheme X is not coded in this function, the site will be given polarity 0, which the code interprets as having cross-interaction with all sites on any different component.
- subroutine **check_site_and_scheme** (site, scheme)

  Sanity check for the input (site,scheme).
- character(len=10) function **get_assoc_string** (assoc)

Variables

- integer, parameter assoc_scheme_1 = 1

  Association schemes. ! NONZERO INTERACTIONS FOR SCHEMES:
- integer, parameter **assoc_scheme_2a** = 2

  AA AB BB.
- integer, parameter **assoc_scheme_2b** = 3

  AB.
- integer, parameter **assoc_scheme_2c** = 4

  AA AB, where A is a bipolar site that associates with all other sites.
- integer, parameter **assoc_scheme_3a** = 5

  AA AB AC BB BC CC.
- integer, parameter **assoc_scheme_3b** = 6

  AC BC.
- integer, parameter **assoc_scheme_4a** = 7

  AA AB AC AD BB BC BD CC CD DD.
- integer, parameter **assoc_scheme_4b** = 8

  AD BD CD.
- integer, parameter **assoc_scheme_4c** = 9

  AC AD BC BD.
- integer, parameter **assoc_scheme_1ea** = 10

  One site with positive polarity.
- integer, parameter **no_assoc** = -1

  No self-association.
- integer, parameter aricomb = 1

  Combining rules.
- integer, parameter **geocomb** = 0

  Geometric mean combining rule. defaultComb is used to select hard coded combining rule. When adding ability to select combining rule for PC-SAFT, defaultComb was added to avoid breaking code that relies on the hard coded combining rules). So any entry in PC-SAFT.json that does not specify eps_comb_rule or beta_comb_rule will be given defaultComb.
- integer, parameter **defaultcomb** = 2
- integer, parameter **nositesflag** = -1

5.3.1 Detailed Description

Functionality for working with association schemes. The work mostly consists of keeping track of indices.

5.3.2 Function/Subroutine Documentation

5.3.2.1 compidx_to_sites()

```
subroutine assocschemeutils::compidx_to_sites (
             type(association), intent(in) assoc,
             integer, intent(in) ic,
             integer, intent(out) k_first,
             integer, intent(out) k_last )
```

Gives the first and last association sites, k_first and k_last, for comp i.

Parameters

in	*ic*	the original component index
out	*k_first*	the first association site number for comp i
out	*k_last*	the last association site number for comp i

5.3.2.2 cross_site_interaction()

```
logical function assocschemeutils::cross_site_interaction (
            integer, intent(in) site1,
            integer, intent(in) site2,
            integer, intent(in) assoc_scheme_i,
            integer, intent(in) assoc_scheme_ii )
```

Given two associating components with different association schemes, one sometimes wants to model only the realistic site–site combinations on the different molecules, e.g. only the positively charged sites on ethanol should interact only with the negatively charged sites on H2O, and vice versa for negatively charged sites on ethanol and positively charged sites on H2O. This is the function which specifies this. It returns true if there is nonzero interaction between site1 on a molecule using scheme I and site2 on a molecule with scheme II.

Parameters

in	*site2*	Interaction sites.
in	*assoc_scheme_ii*	Association schemes.

5.3.2.3 site_interaction_internal()

```
logical function assocschemeutils::site_interaction_internal (
            integer, intent(in) site1,
            integer, intent(in) site2,
            integer, intent(in) assoc_scheme )
```

Returns true if there is nonzero interaction between site1 and site2, according to association scheme assoc_scheme.

Parameters

in	*site2*	Interaction sites
in	*assoc_scheme*	Association scheme

5.3.2.4 site_to_compidx()

```
integer function assocschemeutils::site_to_compidx (
            type(association), intent(in) assoc,
            integer, intent(in) k )
```

Gives the component number ic to which association site number k belongs.

Parameters

in	*k*	the association site number

5.3.3 Variable Documentation

5.3.3.1 aricomb

```
integer, parameter assocschemeutils::aricomb = 1
```

Combining rules.

Arithmetic mean combining rule.

5.3.3.2 assoc_scheme_1

```
integer, parameter assocschemeutils::assoc_scheme_1 = 1
```

Association schemes. ! NONZERO INTERACTIONS FOR SCHEMES:

AA

5.4 cbalpha Module Reference

Routines for setting and computing (T-dependent) alpha correlations in cubic EoS.

Functions/Subroutines

- subroutine, public setsinglealphacorr (i, cbeos, alphaidx, alphaparams)

  Set alpha correlation and parameters for a component. If one wants to use value of alphaParams from the database, one has to retrieve them first using the appropriate get function, e.g. getAlphaMcParamsFromDb.
- subroutine, public getsinglealphacorr (i, cbeos, corrname, numparam, alphaparams)

  Get active alpha correlation parameters.
- subroutine, public tpinitalphacorr (nc, comp, cbeos, alphastr, alpha_reference)

  Initializes alpha correlations (the same correlation for all components).
- subroutine, public **cbcalcalphaterm** (nc, cbeos, t)

  Calculates the alpha term for all components.
- subroutine, public **getacentricalphaparam** (alpha_idx, acfi, params)

5.4.1 Detailed Description

Routines for setting and computing (T-dependent) alpha correlations in cubic EoS.

5.4.2 Function/Subroutine Documentation

5.4.2.1 getsinglealphacorr()

```
subroutine, public cbalpha::getsinglealphacorr (
             integer, intent(in) i,
             class(cb_eos), intent(inout) cbeos,
             character (len=*), intent(in) corrname,
             integer, intent(in) numparam,
             real, dimension(numparam), intent(out) alphaparams )
```

Get active alpha correlation parameters.

Author

Ailo 19.01.16.

5.4.2.2 setsinglealphacorr()

```
subroutine, public cbalpha::setsinglealphacorr (
             integer, intent(in) i,
             class(cb_eos), intent(inout) cbeos,
             integer, intent(in) alphaidx,
             real, dimension(:), intent(in) alphaparams )
```

Set alpha correlation and parameters for a component. If one wants to use value of alphaParams from the database, one has to retrieve them first using the appropriate get function, e.g. getAlphaMcParamsFromDb.

Author

Ailo 19.01.16.

5.4.2.3 tpinitalphacorr()

```
subroutine, public cbalpha::tpinitalphacorr (
             integer, intent(in) nc,
             type(gendata_pointer), dimension(nc), intent(in) comp,
             class(cb_eos), intent(inout) cbeos,
             character(len=*), intent(in) alphastr,
             character(len=*), intent(in), optional alpha_reference )
```

Initializes alpha correlations (the same correlation for all components).

This routine is invoked after eoslibinit as one of the subsequent initialization calls. To change correlation, one has to call setSingleAlphaCorr.

Author

Ailo Aasen

5.5 cbbeta Module Reference

Functionality for temperature-dependent covolume factor beta in cubic EoS, defined by b(T) = b(Tc)∗beta(T)

Functions/Subroutines

- subroutine, public tpinitbetacorr (nc, comp, cbeos, betastr, setno)

  Initializes beta correlations (the same correlation for all components).
- subroutine, public **setsinglebetacorr** (i, cbeos, corrname, betaparams)

  Set beta correlation and parameters for a component.
- subroutine, public **getsinglebetacorr** (i, cbeos, corrname, numparam, betaparams)

  Get active beta correlation parameters.
- subroutine, public **cbcalcbetaterm** (nc, cbeos, t)

  Calculates the beta term for all components.

5.5.1 Detailed Description

Functionality for temperature-dependent covolume factor beta in cubic EoS, defined by b(T) = b(Tc)∗beta(T)

Author

Ailo, Jan 2020

5.5.2 Function/Subroutine Documentation

5.5.2.1 tpinitbetacorr()

```
subroutine, public cbbeta::tpinitbetacorr (
             integer, intent(in) nc,
             type (gendata_pointer), dimension(nc), intent(in) comp,
             class(cb_eos), intent(inout) cbeos,
             character (len=*), intent(in) betastr,
             integer, intent(in), optional setno )
```

Initializes beta correlations (the same correlation for all components).

This routine is invoked after eoslibinit as one of the subsequent initialization calls. To change correlation, one has to call setSingleBetaCorr.

5.6 cbhelm Module Reference

Get reduced Helmholtz function (F = A^r/RT) and differentials from eoscubic type. The cubiceos instance stores F and its derivatives as a function of certain explicit quantities, which includes the variable T, v and n, but also the cubic EoS parameters a and b that depend on (T,V,n). To get the total derivatives of F wrt (T,V,n), these have to be combined in the appropriate way, which is the purpose of this module.

Functions/Subroutines

- real function, public cbf (cubiceos)

  Get reduced residual Helmholtz energy for cubic equation of state.
- real function, public cbfv (cubiceos)

  Get reduced residual Helmholtz energy differential wrpt. specific volume for cubic equation of state.
- real function, public cbfvv (cubiceos)

  Get reduced residual Helmholtz energy second differential wrpt. specific volume for cubic equation of state.
- real function, public cbfvvv (cubiceos, t, v)

  Get reduced residual Helmholtz energy third differential wrpt. specific volume for cubic equation of state.
- real function, public cbft (cubiceos)

  Get reduced residual Helmholtz energy differential wrpt. temperature for cubic equation of state.
- real function, public cbftt (cubiceos)

  Get reduced residual Helmholtz energy second differential wrpt. specific volume for cubic equation of state.
- real function, public cbfvt (cubiceos)

  Get reduced residual Helmholtz energy differential wrpt. specific volume and temperature for cubic equation of state.
- subroutine, public cbfi (nc, cubiceos, fi)

  Get reduced residual Helmholtz energy differentials wrpt. mole numbers for cubic equation of state.
- subroutine, public cbfij (nc, cubiceos, fij)

  Get reduced residual Helmholtz energy second differentials wrpt. mole numbers for cubic equation of state.
- subroutine, public cbfit (nc, cubiceos, fit)

  Get reduced residual Helmholtz energy second differentials wrpt. mole numbers and temperature for cubic equation of state.
- subroutine, public cbfiv (nc, cubiceos, fiv)

  Get reduced residual Helmholtz energy second differentials wrpt. mole numbers and volume for cubic equation of state.
- real function, public cbpress (cubiceos, t, v)

  Pressure from cubic equation of state.
- real function, public cbpv (cubiceos)

  Pressure differential wrpt. volume from cubic equation of state.
- real function, public cbpvv (cubiceos, t, v)

  Second pressure differential wrpt. volume from cubic equation of state.
- real function, public cbprst (cubiceos)

  Pressure differential wrpt. temperature from cubic equation of state.
- subroutine, public cbpi (nc, cubiceos, pi)

  Get pressure differential wrpt. mole number i for cubic equation of state.

5.6.1 Detailed Description

Get reduced Helmholtz function (F = A^r/RT) and differentials from eoscubic type. The cubiceos instance stores F and its derivatives as a function of certain explicit quantities, which includes the variable T, v and n, but also the cubic EoS parameters a and b that depend on (T,V,n). To get the total derivatives of F wrt (T,V,n), these have to be combined in the appropriate way, which is the purpose of this module.

5.6.2 Function/Subroutine Documentation

5.6.2.1 cbf()

```
real function, public cbhelm::cbf (
            class(cb_eos), intent(in) cubiceos )
```

Get reduced residual Helmholtz energy for cubic equation of state.

Author

MH, 2013-11-30

5.6.2.2 cbfi()

```
subroutine, public cbhelm::cbfi (
             integer, intent(in) nc,
             class(cb_eos), intent(in) cubiceos,
             real, dimension(nc), intent(out) fi )
```

Get reduced residual Helmholtz energy differentials wrpt. mole numbers for cubic equation of state.

Author

MH, 2013-11-30

5.6.2.3 cbfij()

```
subroutine, public cbhelm::cbfij (
             integer, intent(in) nc,
             class(cb_eos), intent(in) cubiceos,
             real, dimension(nc,nc), intent(out) fij )
```

Get reduced residual Helmholtz energy second differentials wrpt. mole numbers for cubic equation of state.

Author

MH, 2013-11-30

5.6.2.4 cbfit()

```
subroutine, public cbhelm::cbfit (
             integer, intent(in) nc,
             class(cb_eos), intent(in) cubiceos,
             real, dimension(nc), intent(out) fit )
```

Get reduced residual Helmholtz energy second differentials wrpt. mole numbers and temperature for cubic equation of state.

Author

MH, 2013-11-30

5.6.2.5 cbfiv()

```
subroutine, public cbhelm::cbfiv (
             integer, intent(in) nc,
             class(cb_eos), intent(in) cubiceos,
             real, dimension(nc), intent(out) fiv )
```

Get reduced residual Helmholtz energy second differentials wrpt. mole numbers and volume for cubic equation of state.

Author

MH, 2013-11-30

5.6.2.6 cbft()

```
real function, public cbhelm::cbft (
             class(cb_eos), intent(in) cubiceos )
```

Get reduced residual Helmholtz energy differential wrpt. temperature for cubic equation of state.

Author

MH, 2013-11-30

5.6.2.7 cbftt()

```
real function, public cbhelm::cbftt (
             class(cb_eos), intent(in) cubiceos )
```

Get reduced residual Helmholtz energy second differential wrpt. specific volume for cubic equation of state.

Author

MH, 2013-11-30

5.6.2.8 cbfv()

```
real function, public cbhelm::cbfv (
             class(cb_eos), intent(in) cubiceos )
```

Get reduced residual Helmholtz energy differential wrpt. specific volume for cubic equation of state.

Author

MH, 2013-11-30

5.6.2.9 cbfvt()

```
real function, public cbhelm::cbfvt (
             class(cb_eos), intent(in) cubiceos )
```

Get reduced residual Helmholtz energy differential wrpt. specific volume and temperature for cubic equation of state.

Author

MH, 2013-11-30

5.6.2.10 cbfvv()

```
real function, public cbhelm::cbfvv (
             class(cb_eos), intent(in) cubiceos )
```

Get reduced residual Helmholtz energy second differential wrpt. specific volume for cubic equation of state.

Author

MH, 2013-11-30

5.6.2.11 cbfvvv()

```
real function, public cbhelm::cbfvvv (
             class(cb_eos), intent(in) cubiceos,
             real, intent(in) t,
             real, intent(in) v )
```

Get reduced residual Helmholtz energy third differential wrpt. specific volume for cubic equation of state.

Author

MH, 2015-10

5.6.2.12 cbpi()

```
subroutine, public cbhelm::cbpi (
             integer, intent(in) nc,
             class(cb_eos), intent(in) cubiceos,
             real, dimension(nc), intent(out) pi )
```

Get pressure differential wrpt. mole number i for cubic equation of state.

Author

MH, 2015-03

5.6.2.13 cbpress()

```
real function, public cbhelm::cbpress (
             class(cb_eos), intent(in) cubiceos,
             real, intent(in) t,
             real, intent(in) v )
```

Pressure from cubic equation of state.

Author

MH, 2014-04

5.6.2.14 cbprst()

```
real function, public cbhelm::cbprst (
             class(cb_eos), intent(in) cubiceos )
```

Pressure differential wrpt. temperature from cubic equation of state.

Author

MH, 2015-03

5.6.2.15 cbpv()

```
real function, public cbhelm::cbpv (
             class(cb_eos), intent(in) cubiceos )
```

Pressure differential wrpt. volume from cubic equation of state.

Author

MH, 2015-02

5.6.2.16 cbpvv()

```
real function, public cbhelm::cbpvv (
             class(cb_eos), intent(in) cubiceos,
             real, intent(in) t,
             real, intent(in) v )
```

Second pressure differential wrpt. volume from cubic equation of state.

Author

MH, 2015-02

5.7 cbselect Module Reference

Selection of components, eos etc.

Functions/Subroutines

- subroutine get_mixing_rule_index (eosidx, mrulestr, mruleidx)

  From mixing rule string get mixing rule index.
- subroutine, public selectcubiceos (nc, comp, cbeos, alphastr, alpha_reference, betastr)

  Selection of equation of state and the mixing rule Data from the eos-database is copied to the global variable cbeos.
- subroutine cbcalcparameters (nc, cbeos)

  Calculates constants for the various cubic EOS.
- subroutine selectmixingrules (nc, comp, cbeos, mrulestr, param_reference, b_exponent)

  Selection of equation of state and the mixing rule Data from the eos-database is copied to the global variable cbeos.
- subroutine initcubictcpcacf (nc, comp, cbeos, tcspec, pcspec, acfspec)

  Initialize Tc, Pc and acentric factor to use in the cubic EoS.
- subroutine redefine_tcpcacf_comp_cubic (j, tcspec, pcspec, acfspec, ierr)

  Redefine the critical temperature, critical pressure, and acentric factor of the cubic EoS.
- subroutine redefine_fallback_tcpcacf (tcspec, pcspec, acfspec)

  Redefine the critical temperature, critical pressure, and acentric factor of the cubic fallback EoS. Can be used to enforce the fallback EoS to have the same Tc, Pc and Acf as the main EoS.
- subroutine, public tpselectinteractionparameters (nc, comp, cbeos, param_reference)

  Selection of binary interacion parameters This routine is called with a default set from tpSelectEOS. If more than one set of interaction parameters are available this set can be retrieved. All interaction parameters are stored in the variable kijdb in the kijdatadb structure. The selected binary interaction paramters are stored in a 2-dimensional array in the global cbeos type.
- integer function getcompindex (nc, comp, compid)

  The function retun the index of the working data record (in the module compdata) for the compoentent that is found.
- real function **getkij** (cbeos, eosid, mruleid, ref, uid1, uid2)
- real function **getlij** (cbeos, eosid, mruleid, ref, uid1, uid2)
- subroutine **getinterdatageij** (mge, eosid, ref, uid1, uid2, indxi, indxj, found)

5.7.1 Detailed Description

Selection of components, eos etc.

5.7.2 Function/Subroutine Documentation

5.7.2.1 cbcalcparameters()

```
subroutine cbselect::cbcalcparameters (
            integer, intent(in) nc,
            class(cb_eos), intent(inout) cbeos )
```

Calculates constants for the various cubic EOS.

Author

Geir Skaugen

Date

2012-06-12

Author

Morten Hammer

Parameters

in,out	*cbeos*	Uses the calculated m1 and m2 for two-param eos

5.7.2.2 get_mixing_rule_index()

```
subroutine cbselect::get_mixing_rule_index (
             integer, intent(in) eosidx,
             character (len=*), intent(in) mrulestr,
             integer, intent(out) mruleidx )
```

From mixing rule string get mixing rule index.

Parameters

eosidx	Index of EOS
mrulestr	The mixing rule as a character string e.g 'Classic'

Return values

mruleidx	Mixing rule index The character strings are case-insensitive

Author

Geir S

Morten Hammer

5.7.2.3 getcompindex()

```
integer function cbselect::getcompindex (
             integer, intent(in) nc,
             type(gendata), dimension(nc), intent(in) comp,
             character(len=*), intent(in) compid )
```

The function retun the index of the working data record (in the module compdata) for the compoentent that is found.

Parameters

compid	Character string for a single component ID.

Return values

idx	The array index of compdb.

5.7.2.4 initcubictcpcacf()

```
subroutine cbselect::initcubictcpcacf (
             integer, intent(in) nc,
             type(gendata_pointer), dimension(nc), intent(in) comp,
             class(cb_eos) cbeos,
             real, dimension(nc), intent(in), optional tcspec,
             real, dimension(nc), intent(in), optional pcspec,
             real, dimension(nc), intent(in), optional acfspec )
```

Initialize Tc, Pc and acentric factor to use in the cubic EoS.

Author

Ailo Aasen

Parameters

in	*tcspec*	Specified critical temperature [K]
in	*pcspec*	Specified critical pressure [Pa]
in	*acfspec*	Specified acentric factor [-]

5.7.2.5 redefine_fallback_tcpcacf()

```
subroutine cbselect::redefine_fallback_tcpcacf (
             real, dimension(nce), intent(in), optional tcspec,
             real, dimension(nce), intent(in), optional pcspec,
             real, dimension(nce), intent(in), optional acfspec )
```

Redefine the critical temperature, critical pressure, and acentric factor of the cubic fallback EoS. Can be used to enforce the fallback EoS to have the same Tc, Pc and Acf as the main EoS.

Author

Ailo Aasen

Parameters

in	*tcspec*	Specified critical temperature [K]
in	*pcspec*	Specified critical pressure [Pa]
in	*acfspec*	Specified acentric factor [-]

5.7.2.6 redefine_tcpcacf_comp_cubic()

```
subroutine cbselect::redefine_tcpcacf_comp_cubic (
             integer, intent(in) j,
             real, intent(in) tcspec,
             real, intent(in) pcspec,
```

```
            real, intent(in) acfspec,
            integer, intent(out) ierr )
```

Redefine the critical temperature, critical pressure, and acentric factor of the cubic EoS.

Author

Morten Hammer

Parameters

in	*j*	Component index
out	*ierr*	Component index
in	*tcspec*	Specified critical temperature [K]
in	*pcspec*	Specified critical pressure [Pa]
in	*acfspec*	Specified acentric factor [-]

5.7.2.7 selectcubiceos()

```
subroutine, public cbselect::selectcubiceos (
            integer, intent(in) nc,
            type(gendata_pointer), dimension(:), intent(in) comp,
            class(cb_eos), intent(inout) cbeos,
            character (len=*), intent(in) alphastr,
            character (len=*), intent(in) alpha_reference,
            character (len=*), intent(in), optional betastr )
```

Selection of equation of state and the mixing rule Data from the eos-database is copied to the global variable cbeos.

Parameters

eosstr	The equation of state as a character string e.g 'SRK' og 'PR'
mrulestr	The mixing rule as a character string e.g 'Classic'

The character strings are case-insensitive

Author

Geir S

Morten Hammer

5.7.2.8 selectmixingrules()

```
subroutine cbselect::selectmixingrules (
            integer, intent(in) nc,
            type(gendata_pointer), dimension(nc), intent(in) comp,
            type(cb_eos) cbeos,
            character (len=*), intent(in) mrulestr,
            character (len=*), intent(in) param_reference,
            real, intent(in), optional b_exponent )
```

Selection of equation of state and the mixing rule Data from the eos-database is copied to the global variable cbeos.

Parameters

eosstr	The equation of state as a character string e.g 'SRK' og 'PR'
mrulestr	The mixing rule as a character string e.g 'Classic'

The character strings are case-insensitive

Author

Geir S

Morten Hammer

5.7.2.9 tpselectinteractionparameters()

```
subroutine, public cbselect::tpselectinteractionparameters (
             integer, intent(in) nc,
             type(gendata_pointer), dimension(nc), intent(in) comp,
             type(cb_eos), intent(inout) cbeos,
             character (len=*), intent(in) param_reference )
```

Selection of binary interacion parameters This routine is called with a default set from tpSelectEOS. If more than one set of interaction parameters are available this set can be retrieved. All interaction parameters are stored in the variable kijdb in the kijdatadb structure. The selected binary interaction paramters are stored in a 2-dimensional array in the global cbeos type.

Parameters

setno	Set no

Author

Geir S

5.8 co2_gibbs Module Reference

Module calculating thermodynamic potentials and differentials for solid CO2.

Functions/Subroutines

- subroutine, public sco2init (sl_tr, g_tr, t_tr, p_tr)

  Initialize module for solid co2 Gibbs energy to same reference levels as another EoS.
- real function, public sco2_gibbs (t, p)

  Calculate gibbs energy for dry-ice. Unit: J/mol.
- real function, public sco2_dgdp (t, p)

  Calculate specific volume of dry-ice. Gibbs differential wrpt. pressure at constant temperature. Unit: m3/mol.
- real function, public sco2_d2gdp2 (t, p)

  Calculate specific volume differential wrpt. pressure of dry-ice. Temperature is held constant Unit: m3/mol/Pa.

- real function, public sco2_dgdt (t, p)

  Calculate Gibbs differential wrpt. temperature. Pressure is held constant Unit: J/mol/K.
- real function, public sco2_d2gdt2 (t, p)

  Calculate Gibbs second differential wrpt. temperature. Pressure is held constant Unit: J/mol/K2.
- real function, public sco2_d2gdtdp (t, p)

  Calculate Gibbs second differential wrpt. temperature and pressure. Unit: J/mol/K/Pa.
- real function, public sco2_specvol (t, p)

  Calculate specific volume of dry-ice. Gibbs differential wrpt. pressure at constant temperature. Unit: m3/mol.
- real function, public sco2_entropy (t, p)

  Calculate specific entropy of dry-ice. Unit: J/mol/K.
- real function, public sco2_enthalpy (t, p)

  Calculate specific enthalpy of dry-ice. Unit: J/mol.
- real function, public sco2_energy (t, p)

  Calculate specific internal energy of dry-ice. Unit: J/mol.
- real function, public sco2_helmholtz (t, p)

  Calculate specific Helmholtz energy of dry-ice. Unit: J/mol.
- real function, public sco2_heat_capacity (t, p)

  Calculate specific heat-capasity of dry-ice at constant pressure. Unit: J/mol/K.
- real function, public sco2_speed_of_sound (t, p)

  Calculate speed of sound. Unit: m/s.

5.8.1 Detailed Description

Module calculating thermodynamic potentials and differentials for solid CO2.

Ref: Equation of State for Solid Carbon Dioxide Based on the Gibbs Free Energy Andreas Jäger and Roland Span Journal of Chemical & Engineering Data (J. Chem. Eng. Data) Pages: 590-597 January 2012 Number 57

5.8.2 Function/Subroutine Documentation

5.8.2.1 sco2_d2gdp2()

```
real function, public co2_gibbs::sco2_d2gdp2 (
            real, intent(in) t,
            real, intent(in) p )
```

Calculate specific volume differential wrpt. pressure of dry-ice. Temperature is held constant Unit: m3/mol/Pa.

Author

MH, 2013-03-01

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure

5.8.2.2 sco2_d2gdt2()

```
real function, public co2_gibbs::sco2_d2gdt2 (
             real, intent(in) t,
             real, intent(in) p )
```

Calculate Gibbs second differential wrpt. temperature. Pressure is held constant Unit: J/mol/K2.

Author

MH, 2013-03-01

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure

5.8.2.3 sco2_d2gdtdp()

```
real function, public co2_gibbs::sco2_d2gdtdp (
             real, intent(in) t,
             real, intent(in) p )
```

Calculate Gibbs second differential wrpt. temperature and pressure. Unit: J/mol/K/Pa.

Author

MH, 2013-03-01

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure

5.8.2.4 sco2_dgdp()

```
real function, public co2_gibbs::sco2_dgdp (
             real, intent(in) t,
             real, intent(in) p )
```

Calculate specific volume of dry-ice. Gibbs differential wrpt. pressure at constant temperature. Unit: m3/mol.

Author

MH, 2013-03-01

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure

5.8.2.5 sco2_dgdt()

```
real function, public co2_gibbs::sco2_dgdt (
             real, intent(in) t,
             real, intent(in) p )
```

Calculate Gibbs differential wrpt. temperature. Pressure is held constant Unit: J/mol/K.

Author

MH, 2013-03-01

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure

5.8.2.6 sco2_energy()

```
real function, public co2_gibbs::sco2_energy (
             real, intent(in) t,
             real, intent(in) p )
```

Calculate specific internal energy of dry-ice. Unit: J/mol.

Author

MH, 2013-03-01

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure

5.8.2.7 sco2_enthalpy()

```
real function, public co2_gibbs::sco2_enthalpy (
             real, intent(in) t,
             real, intent(in) p )
```

Calculate specific enthalpy of dry-ice. Unit: J/mol.

Author

MH, 2013-03-01

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure

5.8.2.8 sco2_entropy()

```
real function, public co2_gibbs::sco2_entropy (
            real, intent(in) t,
            real, intent(in) p )
```

Calculate specific entropy of dry-ice. Unit: J/mol/K.

Author

MH, 2013-03-01

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure

5.8.2.9 sco2_gibbs()

```
real function, public co2_gibbs::sco2_gibbs (
            real, intent(in) t,
            real, intent(in) p )
```

Calculate gibbs energy for dry-ice. Unit: J/mol.

Author

MH, 2013-03-01

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure

5.8.2.10 sco2_heat_capacity()

```
real function, public co2_gibbs::sco2_heat_capacity (
```

```
            real, intent(in) t,
            real, intent(in) p )
```

Calculate specific heat-capasity of dry-ice at constant pressure. Unit: J/mol/K.

Author

MH, 2013-03-01

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure

5.8.2.11 sco2_helmholtz()

```
real function, public co2_gibbs::sco2_helmholtz (
            real, intent(in) t,
            real, intent(in) p )
```

Calculate specific Helmholtz energy of dry-ice. Unit: J/mol.

Author

MH, 2013-03-01

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure

5.8.2.12 sco2_specvol()

```
real function, public co2_gibbs::sco2_specvol (
            real, intent(in) t,
            real, intent(in) p )
```

Calculate specific volume of dry-ice. Gibbs differential wrpt. pressure at constant temperature. Unit: m3/mol.

Author

MH, 2013-03-01

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure

5.8.2.13 sco2_speed_of_sound()

```
real function, public co2_gibbs::sco2_speed_of_sound (
             real, intent(in) t,
             real, intent(in) p )
```

Calculate speed of sound. Unit: m/s.

Author

MH, 2013-03-01

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure

5.8.2.14 sco2init()

```
subroutine, public co2_gibbs::sco2init (
             real, intent(in) sl_tr,
             real, intent(in) g_tr,
             real, intent(in) t_tr,
             real, intent(in) p_tr )
```

Initialize module for solid co2 Gibbs energy to same reference levels as another EoS.

Author

MH, 2013-03-01

Parameters

in	*sl_tr*	J/mol/K - Liquid entropy at triple point
in	*g_tr*	J/mol - Gibbs energy at triple point
in	*t_tr*	K - Triple point temperature
in	*p_tr*	Pa - Triple point pressure

5.9 compdata Module Reference

The module compdata stores pure component data. After initialisation of a mixture, data from the database "compdatadb.f90" are selected and copied into the working array "comp" of active components.

Data Types

- type alphadatadb

  Alpha correlation for cubic EoS.
- type cidatadb

  Volume shift parameters.
- type cpadata

  Pure component parameters. This data structure stores pure component parameters for the CPA-SRK and CPA-PR equations of state.
- type cpdata

  Ideal heat capacity at constant pressure.
- type gendata
- type gendata_pointer
- type gendatadb

Functions/Subroutines

- logical function **iscomponent** (cid, cname)

  Is this component named cname?
- logical function **isref** (ref, ref_list)

  Is referance tag in referance list?
- logical function **iseos** (eosid, eos)

  Is it match in eosid?
- subroutine **assign_gendatadb** (this, cmp)

  Assignment operator for gendatadb.
- subroutine **assign_gendata** (this, cmp)

  Assignment operator for gendata.
- integer function, public **compindex** (complist, compname)
- subroutine, public **initcomplist** (componentstring, ncomp, complist)
- integer function, public **parsecompvector** (compvector)
- integer function, public **getcomp** (compvector)
- subroutine init_component_data_from_db (complist, nc, ref, comp, ierr)

  Initialize component data from compdatadb.
- subroutine, public **deallocate_comp** (comp)
- subroutine, public **copy_comp** (comp_cpy, comp)
- subroutine cidatadb_get_vol_trs_c (cid, t, ci, cit, citt, ci_temp_dep)
- subroutine **cidatadb_set_zero_vol_trs** (cid)

Variables

- integer, parameter **cp_poly3_cal** =1
- integer, parameter **cp_api44_mass** =2
- integer, parameter **cp_hypotetic_mass** =3
- integer, parameter **cp_poly3_si** =4
- integer, parameter **cp_ici_mass** =5
- integer, parameter **cp_chen_bender_mass** =6
- integer, parameter **cp_dippr_kmol** =7
- integer, parameter **cp_poly4_si** =8
- integer, parameter **cp_mogensen_si** =9
- integer, parameter **cp_h2_kmol** =10
- integer, parameter **cp_trend_si** =11
- integer, parameter **cp_shomate_si** =12
- integer, parameter **vs_constant** = 1
- integer, parameter **vs_linear** = 2
- integer, parameter **vs_quadratic** = 3
- integer, parameter **vs_quintic** = 6

5.9.1 Detailed Description

The module compdata stores pure component data. After initialisation of a mixture, data from the database "compdatadb.f90" are selected and copied into the working array "comp" of active components.

5.9.2 Function/Subroutine Documentation

5.9.2.1 cidatadb_get_vol_trs_c()

```
subroutine compdata::cidatadb_get_vol_trs_c (
             class(cidatadb), intent(in) cid,
             real, intent(in) t,
             real, intent(out) ci,
             real, intent(out) cit,
             real, intent(out) citt,
             logical, intent(out) ci_temp_dep )
```

Parameters

in	*t*	Temperature (K)
out	*ci*	Volume translation (m3/mol)
out	*cit*	Volume translation differential (m3/mol/K)
out	*citt*	Volume translation second differential (m3/mol/K2)
out	*ci_temp_dep*	Volume translation is temp. dependent

5.9.2.2 init_component_data_from_db()

```
subroutine compdata::init_component_data_from_db (
             character(len=*), dimension(:), intent(in) complist,
             integer, intent(in) nc,
             character(len=*), intent(in) ref,
             type(gendata_pointer), dimension(:), intent(inout), allocatable comp,
             integer, intent(out) ierr )
```

Initialize component data from compdatadb.

Parameters

in	*complist*	List of component names
in	*nc*	Number of components
in	*ref*	Reference for ideal cp correlation
in,out	*comp*	Pointer to structure for holding data

5.10 compdatadb Module Reference

Automatically generated to file compdatadb.f90 using utility python code pyUtils Time stamp: 2023-09-28T12:56:50.126803.

Variables

- type(gendatadb), parameter **cx1** = gendatadb(ident = "BUT1OL", formula = "C4H10O", name = "1-BUTANOL", mw = 74.1216, Tc = 562.2000, Pc = 4500000.00, Zc = 0.263800, acf = 0.592980, Tb = 390.6000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.254900, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp1** = cpdata(cid = "BUT1OL", ref = "Default", bib_ref = "", cptype = 2, cp = (/0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -175.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu1** = alphadatadb(eosid="PR", cid="BUT1OL", ref="tcPR", coeff=(/1.62670000e+00, 1.00000000e+00, 6.99800000e-01/))
- type(cidatadb), parameter **c1** = cidatadb(eosid="PR", cid="BUT1OL", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=1.91930000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu2** = alphadatadb(eosid="SRK", cid="BUT1OL", ref="tcRK", coeff=(/1.51560000e+00, 1.00000000e+00, 8.55400000e-01/))
- type(cidatadb), parameter **c2** = cidatadb(eosid="SRK", cid="BUT1OL", ref="tcRK", bib_ref="10.1016/j.fluid.2016.09.003", ciA=1.74514000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cpadata), parameter **cpa1** = CPAdata(eosid="CPA-SRK", compName="BUT1OL", ref="Default/Oliveira2008", bib_reference="10.1016/j.fluid.2008.02.020", a0=1.80190000e+06, b=8.13090000e-02, eps=2.00690000e+04, beta=3.66940000e-03, alphacorridx = cbAlphaClassicIdx, alphaParams = (/9.87660000e-01,0.00000000e+00,0.00000000e+00/), assoc_scheme = assoc_scheme_2B)
- type(gendatadb), parameter **cx2** = gendatadb(ident = "HEX1OL", formula = "C6H14O", name = "1-HEXANOL", mw = 102.1748, Tc = 610.5000, Pc = 3413000.00, Zc = 0.260200, acf = 0.570000, Tb = 430.0000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.254400, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp2** = cpdata(cid = "HEX1OL", ref = "Default", bib_ref = "", cptype = 2, cp = (/0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -175.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu3** = alphadatadb(eosid="PR", cid="HEX1OL", ref="tcPR", coeff=(/2.35820000e+00, 1.00000000e+00, 4.48600000e-01/))
- type(cidatadb), parameter **c3** = cidatadb(eosid="PR", cid="HEX1OL", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=4.15340000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu4** = alphadatadb(eosid="SRK", cid="HEX1OL", ref="tcRK", coeff=(/2.08320000e+00, 1.00000000e+00, 5.78300000e-01/))
- type(cidatadb), parameter **c4** = cidatadb(eosid="SRK", cid="HEX1OL", ref="tcRK", bib_ref="10.1016/j.fluid.2016.09.003", ciA=2.59066000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cpadata), parameter **cpa2** = CPAdata(eosid="CPA-SRK", compName="HEX1OL", ref="Default/Oliveira2008", bib_reference="10.1016/j.fluid.2008.02.020", a0=2.83860000e+06, b=1.13130000e-01, eps=2.27590000e+04, beta=1.67270000e-03, alphacorridx = cbAlphaClassicIdx, alphaParams = (/9.69590000e-01,0.00000000e+00,0.00000000e+00/), assoc_scheme = assoc_scheme_2B)
- type(gendatadb), parameter **cx3** = gendatadb(ident = "PENT1OL", formula = "C5H12O", name = "1-PENTANOL", mw = 88.1482, Tc = 580.0000, Pc = 3900000.00, Zc = 0.264300, acf = 0.578980, Tb = 411.0000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.256100, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp3** = cpdata(cid = "PENT1OL", ref = "Default", bib_ref = "", cptype = 2, cp = (/0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -175.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu5** = alphadatadb(eosid="PR", cid="PENT1OL", ref="tcPR", coeff=(/1.68580000e+00, 1.00000000e+00, 6.61800000e-01/))
- type(cidatadb), parameter **c5** = cidatadb(eosid="PR", cid="PENT1OL", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=1.47820000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)

- type(alphadatadb), parameter **twu6** = alphadatadb(eosid="SRK", cid="PENT1OL", ref="tcRK", coeff=(/1.↩
63720000e+00, 1.00000000e+00, 7.71700000e-01/))
- type(cidatadb), parameter **c6** = cidatadb(eosid="SRK", cid="PENT1OL", ref="tcRK", bib_ref="10.1016/j.↩
fluid.2016.09.003", ciA=1.99375000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cpadata), parameter **cpa3** = CPAdata(eosid="CPA-SRK", compName="PENT1OL", ref="Default/Oliveira2008",
bib_reference="10.1016/j.fluid.2008.02.020", a0=2.35520000e+06, b=9.71790000e-02, eps=1.86660000e+04,
beta=2.67240000e-03, alphacorridx = cbAlphaClassicIdx, alphaParams = (/1.06900000e+00,0.↩
00000000e+00,0.00000000e+00/), assoc_scheme = assoc_scheme_2B)
- type(gendatadb), parameter **cx4** = gendatadb(ident = "PROP1OL", formula = "C3H8O", name = "1-
PROPANOL", mw = 60.0950, Tc = 536.9000, Pc = 5200000.00, Zc = 0.254300, acf = 0.623000, Tb = 370.↩
3000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1,
ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra =
0.251100, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp4** = cpdata(cid = "PROP1OL", ref = "Default", bib_ref = "", cptype =
2, cp = (/0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00, 0.↩
00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -175.↩
0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu7** = alphadatadb(eosid="PR", cid="PROP1OL", ref="tcPR", coeff=(/1.↩
37950000e+00, 1.00000000e+00, 8.77200000e-01/))
- type(cidatadb), parameter **c7** = cidatadb(eosid="PR", cid="PROP1OL", ref="tcPR", bib_ref="10.1016/j.↩
fluid.2016.09.003", ciA=3.27770000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu8** = alphadatadb(eosid="SRK", cid="PROP1OL", ref="tcRK", coeff=(/1.↩
37350000e+00, 1.00000000e+00, 9.94100000e-01/))
- type(cidatadb), parameter **c8** = cidatadb(eosid="SRK", cid="PROP1OL", ref="tcRK", bib_ref="10.1016/j.↩
fluid.2016.09.003", ciA=1.58803000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cpadata), parameter **cpa4** = CPAdata(eosid="CPA-SRK", compName="PROP1OL", ref="Default/Oliveira2008",
bib_reference="10.1016/j.fluid.2008.02.020", a0=1.14240000e+06, b=6.37880000e-02, eps=2.19130000e+04,
beta=7.73600000e-03, alphacorridx = cbAlphaClassicIdx, alphaParams = (/9.01340000e-01,0.↩
00000000e+00,0.00000000e+00/), assoc_scheme = assoc_scheme_2B)
- type(gendatadb), parameter **cx5** = gendatadb(ident = "13BD", formula = "C4H6", name = "1,3-BUTADIENE",
mw = 54.0920, Tc = 425.0000, Pc = 4330000.00, Zc = 0.270000, acf = 0.195000, Tb = 268.7000, Ttr =
0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.↩
00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.267500,
mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp5** = cpdata(cid = "13BD", ref = "Default", bib_ref = "", cptype = 4, cp = (/-1.↩
68700000e+00,3.41900000e-01,-2.34000000e-04,6.33500000e-08,0.00000000e+00, 0.00000000e+00,0.↩
00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000
)
- type(alphadatadb), parameter **twu9** = alphadatadb(eosid="PR", cid="13BD", ref="tcPR", coeff=(/1.↩
61800000e-01, 8.59200000e-01, 2.40880000e+00/))
- type(cidatadb), parameter **c9** = cidatadb(eosid="PR", cid="13BD", ref="tcPR", bib_ref="10.1016/j.fluid.↩
2016.09.003", ciA=-1.61400000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu10** = alphadatadb(eosid="SRK", cid="13BD", ref="tcRK", coeff=(/2.↩
22300000e-01, 8.58400000e-01, 2.41930000e+00/))
- type(cidatadb), parameter **c10** = cidatadb(eosid="SRK", cid="13BD", ref="tcRK", bib_ref="10.1016/j.fluid.↩
2016.09.003", ciA=1.12082000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx6** = gendatadb(ident = "2MHX", formula = "C7H16", name = "2-
METHYLHEXANE", mw = 100.2050, Tc = 530.4000, Pc = 2730000.00, Zc = 0.261000, acf = 0.329000,
Tb = 363.2000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000,
psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax =
0.0000, Zra = 0.263800, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp6** = cpdata(cid = "2MHX", ref = "Default", bib_ref = "", cptype = 2, cp = (/1.↩
78937090e+01,4.04849000e-01,1.33465300e-03,2.87769800e-06,-3.51181800e-09, 1.25400550e-12,1.↩
82345600e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -75.0000, Tcpmax = 700.↩
0000)
- type(alphadatadb), parameter **twu11** = alphadatadb(eosid="PR", cid="2MHX", ref="tcPR", coeff=(/3.↩
79600000e-01, 8.10300000e-01, 1.68300000e+00/))

- type(cidatadb), parameter **c11** = cidatadb(eosid="PR", cid="2MHX", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=7.27700000e-07, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu12** = alphadatadb(eosid="SRK", cid="2MHX", ref="tcRK", coeff=(/3.↩ 42600000e-01, 8.32200000e-01, 2.20210000e+00/))
- type(cidatadb), parameter **c12** = cidatadb(eosid="SRK", cid="2MHX", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=2.51160000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx7** = gendatadb(ident = "3MP", formula = "C6H14", name = "3-METHYLPENTANE", mw = 86.1780, Tc = 504.5000, Pc = 3120000.00, Zc = 0.273000, acf = 0.272000, Tb = 336.4000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.269000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp7** = cpdata(cid = "3MP", ref = "Default", bib_ref = "", cptype = 2, cp = (/1.↩ 79647680e+01,3.97799000e-01,1.20987000e-03,3.25455600e-06,-3.94266100e-09, 1.43841480e-12,2.↩ 14954100e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -75.0000, Tcpmax = 700.↩ 0000)
- type(alphadatadb), parameter **twu13** = alphadatadb(eosid="PR", cid="3MP", ref="tcPR", coeff=(/2.↩ 52500000e-01, 8.32200000e-01, 2.04680000e+00/))
- type(cidatadb), parameter **c13** = cidatadb(eosid="PR", cid="3MP", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-1.74790000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu14** = alphadatadb(eosid="SRK", cid="3MP", ref="tcRK", coeff=(/2.↩ 76300000e-01, 8.45100000e-01, 2.37500000e+00/))
- type(cidatadb), parameter **c14** = cidatadb(eosid="SRK", cid="3MP", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.88966000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx8** = gendatadb(ident = "ACETONE", formula = "C3H6O", name = "ACETONE", mw = 58.0800, Tc = 508.1000, Pc = 4700000.00, Zc = 0.233000, acf = 0.307000, Tb = 329.2200, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.↩ 00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.244200, mu_dipole = 2.880000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp8** = cpdata(cid = "ACETONE", ref = "Default", bib_ref = "", cptype = 6, cp = (/7.↩ 33799960e-01,2.16303500e-04,8.20407250e-06,-1.02740600e-08,3.90520150e-12, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 50.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu15** = alphadatadb(eosid="PR", cid="ACETONE", ref="tcPR", coeff=(/6.↩ 04100000e-01, 8.40200000e-01, 1.19840000e+00/))
- type(cidatadb), parameter **c15** = cidatadb(eosid="PR", cid="ACETONE", ref="tcPR", bib_ref="10.1016/j.↩ fluid.2016.09.003", ciA=1.26537000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu16** = alphadatadb(eosid="SRK", cid="ACETONE", ref="tcRK", coeff=(/3.↩ 20700000e-01, 8.54700000e-01, 2.36300000e+00/))
- type(cidatadb), parameter **c16** = cidatadb(eosid="SRK", cid="ACETONE", ref="tcRK", bib_ref="10.1016/j.↩ fluid.2016.09.003", ciA=2.61512000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx9** = gendatadb(ident = "ACETYLENE", formula = "C2H2", name = "ACETYLENE", mw = 26.0380, Tc = 308.3000, Pc = 6140000.00, Zc = 0.270000, acf = 0.190000, Tb = 188.4000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.↩ 00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = -1.000000, mu_dipole = 0.000000, q_quadrupole = 4.600000)
- type(cpdata), parameter **cp9** = cpdata(cid = "ACETYLENE", ref = "Default", bib_ref = "", cptype = 4, cp = (/2.↩ 68200000e+01,7.57800000e-02,-5.00700000e-05,1.41200000e-08,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(gendatadb), parameter **cx10** = gendatadb(ident = "ALLENE", formula = "C3H4", name = "PROPADIENE", mw = 40.0650, Tc = 393.8500, Pc = 5248600.00, Zc = 0.259700, acf = 0.120000, Tb = 238.6500, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/6.53610000e+00, 1.05472000e+03, -7.70800000e+01/), Tantmin = 174.0000, Tantmax = 257.0000, Zra = 0.267700, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp10** = cpdata(cid = "ALLENE", ref = "Default", bib_ref = "", cptype = 4, cp = (/9.↩ 90600000e+00,1.97700000e-01,-1.18200000e-04,2.78200000e-08,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)

- type(alphadatadb), parameter **twu17** = alphadatadb(eosid="PR", cid="ALLENE", ref="tcPR", coeff=(/1.08530000e+00, 4.78800000e-01, 4.59300000e-01/))
- type(cidatadb), parameter **c17** = cidatadb(eosid="PR", cid="ALLENE", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=1.92300000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu18** = alphadatadb(eosid="SRK", cid="ALLENE", ref="tcRK", coeff=(/4.19800000e-01, 7.25600000e-01, 1.03070000e+00/))
- type(cidatadb), parameter **c18** = cidatadb(eosid="SRK", cid="ALLENE", ref="tcRK", bib_ref="10.1016/j.fluid.2016.09.003", ciA=1.18997000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx11** = gendatadb(ident = "NH3", formula = "NH3", name = "AMMONIA", mw = 17.0310, Tc = 405.6000, Pc = 11470000.00, Zc = 0.242000, acf = 0.250000, Tb = 239.7000, Ttr = 0.0000, Ptr = 0.0000, sref = 192.7700, href = -45900.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.69481000e+01, 2.13250000e+03, -3.29800000e+01/), Tantmin = 179.0000, Tantmax = 261.0000, Zra = 0.246500, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp11** = cpdata(cid = "NH3", ref = "Default", bib_ref = "", cptype = 2, cp = (/-2.20260600e+00,2.01031700e+00,-6.50061000e-04,2.37326400e-06,-1.59759500e-09, 3.76173900e-13,9.90447000e-01,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -175.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu19** = alphadatadb(eosid="PR", cid="NH3", ref="tcPR", coeff=(/2.27400000e-01, 8.64500000e-01, 2.33200000e+00/))
- type(alphadatadb), parameter **mc1** = alphadatadb(eosid="PR", cid="NH3", ref="Chapoy2005", coeff=(/7.48000000e-01, -2.50000000e-02, 1.00000000e-03/))
- type(cidatadb), parameter **c19** = cidatadb(eosid="PR", cid="NH3", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=4.03030000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu20** = alphadatadb(eosid="SRK", cid="NH3", ref="tcRK", coeff=(/2.98100000e-01, 8.65100000e-01, 2.32450000e+00/))
- type(alphadatadb), parameter **mc2** = alphadatadb(eosid="SRK", cid="NH3", ref="Chapoy2005", coeff=(/9.16000000e-01, -3.69000000e-01, 4.17000000e-01/))
- type(cidatadb), parameter **c20** = cidatadb(eosid="SRK", cid="NH3", ref="tcRK", bib_ref="10.1016/j.fluid.2016.09.003", ciA=8.63980000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cpadata), parameter **cpa5** = CPAdata(eosid="CPA-SRK", compName="NH3", ref="Default/SINTEF", bib_reference="", a0=3.73160000e+05, b=2.07666000e-02, eps=7.60835000e+03, beta=7.93725000e-04, alphacorridx = cbAlphaClassicIdx, alphaParams = (/7.17324000e-01,0.00000000e+00,0.00000000e+00/), assoc_scheme = assoc_scheme_2B)
- type(gendatadb), parameter **cx12** = gendatadb(ident = "AR", formula = "AR", name = "ARGON", mw = 39.9480, Tc = 150.8000, Pc = 4873700.00, Zc = 0.291000, acf = -0.004000, Tb = 87.3000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.52330000e+01, 7.00510000e+02, -5.84000000e+00/), Tantmin = 81.0000, Tantmax = 94.0000, Zra = 0.308500, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp12** = cpdata(cid = "AR", ref = "Default", bib_ref = "", cptype = 1, cp = (/4.96900000e+00,-7.67000000e-06,1.23400000e-08,0.00000000e+00,0.00000000e+00, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu21** = alphadatadb(eosid="PR", cid="AR", ref="tcPR", coeff=(/1.22800000e-01, 9.04500000e-01, 1.85390000e+00/))
- type(alphadatadb), parameter **mc3** = alphadatadb(eosid="PR", cid="AR", ref="Default", coeff=(/3.97483000e-01, -2.82393000e-01, 7.96288000e-01/))
- type(cidatadb), parameter **c21** = cidatadb(eosid="PR", cid="AR", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=-3.29390000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu22** = alphadatadb(eosid="SRK", cid="AR", ref="tcRK", coeff=(/2.02300000e-01, 9.08600000e-01, 1.81290000e+00/))
- type(cidatadb), parameter **c22** = cidatadb(eosid="SRK", cid="AR", ref="tcRK", bib_ref="10.1016/j.fluid.2016.09.003", ciA=8.72800000e-07, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx13** = gendatadb(ident = "BENZENE", formula = "C6H6", name = "BENZENE", mw = 78.1140, Tc = 562.1000, Pc = 4894000.00, Zc = 0.271000, acf = 0.212000, Tb = 353.3000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.59008000e+01, 2.78851000e+03, -5.23600000e+01/), Tantmin = 280.0000, Tantmax = 377.0000, Zra = 0.269800, mu_dipole = 0.000000, q_quadrupole = 0.000000)

- type(cpdata), parameter **cp13** = cpdata(cid = "BENZENE", ref = "Default", bib_ref = "", cptype = 2, cp = (/8.44670620e+01,-5.13560000e-01,3.24874000e-03,-1.54391300e-06,3.65037000e-10, -2.48222000e-14,5.63104100e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -20.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **mc4** = alphadatadb(eosid="PR", cid="BENZENE", ref="Chapoy2005", coeff=(/7.01000000e-01, -2.52000000e-01, 9.76000000e-01/))
- type(alphadatadb), parameter **twu23** = alphadatadb(eosid="PR", cid="BENZENE", ref="tcPR", coeff=(/1.↩ 26100000e-01, 8.46000000e-01, 2.61370000e+00/))
- type(cidatadb), parameter **c23** = cidatadb(eosid="PR", cid="BENZENE", ref="tcPR", bib_ref="10.1016/j.↩ fluid.2016.09.003", ciA=-1.39140000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **mc5** = alphadatadb(eosid="SRK", cid="BENZENE", ref="Chapoy2005", coeff=(/8.40000000e-01, -3.89000000e-01, 9.17000000e-01/))
- type(alphadatadb), parameter **twu24** = alphadatadb(eosid="SRK", cid="BENZENE", ref="tcRK", coeff=(/1.↩ 82300000e-01, 8.44800000e-01, 2.63540000e+00/))
- type(cidatadb), parameter **c24** = cidatadb(eosid="SRK", cid="BENZENE", ref="tcRK", bib_ref="10.1016/j.↩ fluid.2016.09.003", ciA=1.35115000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx14** = gendatadb(ident = "BUTANAL", formula = "C4H8O", name = "BUTANAL", mw = 72.1070, Tc = 537.2000, Pc = 4410000.00, Zc = 0.249000, acf = 0.282600, Tb = 348.0000, Ttr = 176.8000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.↩ 00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.259300, mu_dipole = 2.720000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp14** = cpdata(cid = "BUTANAL", ref = "Default", bib_ref = "None", cptype = 100, cp = (/0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00, 0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu25** = alphadatadb(eosid="PR", cid="BUTANAL", ref="tcPR", coeff=(/1.↩ 97600000e-01, 8.50000000e-01, 2.54930000e+00/))
- type(cidatadb), parameter **c25** = cidatadb(eosid="PR", cid="BUTANAL", ref="tcPR", bib_ref="Le Guennec et al. (2016): 10.1016/j.fluid.2016.09.003", ciA=4.04360000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu26** = alphadatadb(eosid="SRK", cid="BUTANAL", ref="tcRK", coeff=(/2.↩ 81600000e-01, 8.54400000e-01, 2.47990000e+00/))
- type(cidatadb), parameter **c26** = cidatadb(eosid="SRK", cid="BUTANAL", ref="tcRK", bib_ref="Le Guennec et al. (2016): 10.1016/j.fluid.2016.09.003", ciA=1.95438000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx15** = gendatadb(ident = "CO2", formula = "CO2", name = "CARBON DIOXIDE", mw = 44.0100, Tc = 304.2000, Pc = 7376500.00, Zc = 0.274000, acf = 0.225000, Tb = 194.7000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/2.25898000e+01, 3.10339000e+03, -1.60000000e-01/), Tantmin = 154.0000, Tantmax = 204.0000, Zra = 0.272200, mu_dipole = 0.000000, q_quadrupole = 4.400000)
- type(cpdata), parameter **cp15** = cpdata(cid = "CO2", ref = "Default", bib_ref = "", cptype = 2, cp = (/1.↩ 11137440e+01,4.79107000e-01,7.62159000e-04,-3.59392000e-07,8.47440000e-11, -5.77520000e-15,2.↩ 71918000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -175.0000, Tcpmax = 1200.0000)
- type(cpdata), parameter **cp16** = cpdata(cid = "CO2", ref = "", bib_ref = "", cptype = 5, cp = (/5.↩ 67998955e-01,1.25397835e-03,-7.65547666e-07,1.80606165e-10,-3.10473875e+03, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 50.0000, Tcpmax = 5000.↩ 0000)
- type(alphadatadb), parameter **twu27** = alphadatadb(eosid="PR", cid="CO2", ref="tcPR", coeff=(/1.↩ 78300000e-01, 8.59000000e-01, 2.41070000e+00/))
- type(alphadatadb), parameter **mc6** = alphadatadb(eosid="PR", cid="CO2", ref="Default", coeff=(/7.↩ 04606000e-01, -3.14862000e-01, 1.89083000e+00/))
- type(alphadatadb), parameter **mc7** = alphadatadb(eosid="PR", cid="CO2", ref="Chapoy2005", coeff=(/7.↩ 05000000e-01, -3.15000000e-01, 1.89000000e+00/))
- type(cidatadb), parameter **c27** = cidatadb(eosid="PR", cid="CO2", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-1.13680000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)

- type(alphadatadb), parameter **twu28** = alphadatadb(eosid="SRK", cid="CO2", ref="tcRK", coeff=(/2.80700000e-01, 8.68500000e-01, 2.27780000e+00/))
- type(alphadatadb), parameter **mc8** = alphadatadb(eosid="SRK", cid="CO2", ref="Chapoy2005", coeff=(/8.67000000e-01, -6.74000000e-01, 2.47100000e+00/))
- type(cidatadb), parameter **c28** = cidatadb(eosid="SRK", cid="CO2", ref="tcRK", bib_ref="10.1016/j.fluid.2016.09.003", ciA=4.15820000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cpadata), parameter **cpa6** = CPAdata(eosid="CPA-SRK", compName="CO2", ref="Default/Kontogeorgis-Folas2010", bib_reference="10.1002/9780470747537", a0=3.50790000e+05, b=2.72000000e-02, eps=0.00000000e+00, beta=0.00000000e+00, alphacorridx = cbAlphaClassicIdx, alphaParams = (/7.60200000e-01,0.00000000e+00,0.00000000e+00/), assoc_scheme = no_assoc)
- type(cpadata), parameter **cpa7** = CPAdata(eosid="CPA-SRK", compName="CO2", ref="SINTEF", bib_reference="", a0=3.50790000e+05, b=2.72000000e-02, eps=0.00000000e+00, beta=5.00000000e-02, alphacorridx = cbAlphaClassicIdx, alphaParams = (/7.60200000e-01,0.00000000e+00,0.00000000e+00/), assoc_scheme = assoc_scheme_1EA)
- type(cpadata), parameter **cpa8** = CPAdata(eosid="CPA-SRK", compName="CO2", ref="SINTEF2", bib_reference="", a0=2.99377508e+05, b=2.68729432e-02, eps=1.28604049e+04, beta=9.08545617e-03, alphacorridx = cbAlphaClassicIdx, alphaParams = (/4.62138711e-01,0.00000000e+00,0.00000000e+00/), assoc_scheme = assoc_scheme_1)
- type(gendatadb), parameter **cx16** = gendatadb(ident = "CO", formula = "CO", name = "CARBON MONOXIDE", mw = 28.0100, Tc = 132.8500, Pc = 3494000.00, Zc = 0.292000, acf = 0.045000, Tb = 81.6600, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.54140000e+01, 6.71763100e+02, -5.15400000e+00/), Tantmin = 69.7300, Tantmax = 88.0800, Zra = 0.289600, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp17** = cpdata(cid = "CO", ref = "Default", bib_ref = "", cptype = 6, cp = (/1.16120000e+00,-1.16150000e-03,3.50860000e-06,-3.86480000e-09,1.52870000e-12, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 50.0000, Tcpmax = 1000.0000)
- type(alphadatadb), parameter **twu29** = alphadatadb(eosid="PR", cid="CO", ref="tcPR", coeff=(/9.83000000e-02, 8.77700000e-01, 2.15680000e+00/))
- type(alphadatadb), parameter **mc9** = alphadatadb(eosid="PR", cid="CO", ref="Default", coeff=(/7.05000000e-01, -3.18500000e-01, 1.90120000e+00/))
- type(cidatadb), parameter **c29** = cidatadb(eosid="PR", cid="CO", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=-3.67610000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu30** = alphadatadb(eosid="SRK", cid="CO", ref="tcRK", coeff=(/1.62500000e-01, 8.77800000e-01, 2.15620000e+00/))
- type(cidatadb), parameter **c30** = cidatadb(eosid="SRK", cid="CO", ref="tcRK", bib_ref="10.1016/j.fluid.2016.09.003", ciA=1.40680000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx17** = gendatadb(ident = "CL2", formula = "CL2", name = "CHLORINE", mw = 70.9050, Tc = 417.1500, Pc = 7710000.00, Zc = 0.275000, acf = 0.068800, Tb = 239.1200, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/4.06280000e+00, 8.61340000e+02, -2.68200000e+01/), Tantmin = 176.3100, Tantmax = 255.7900, Zra = 0.277400, mu_dipole = 0.000000, q_quadrupole = 4.100000)
- type(cpdata), parameter **cp18** = cpdata(cid = "CL2", ref = "Default", bib_ref = "Poling et al. (2001): 978-0-07-011682-5", cptype = 8, cp = (/3.05600000e+00,5.37080000e-03,-8.09800000e-06,5.69300000e-09,-1.52560000e-12, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 50.0000, Tcpmax = 1000.0000)
- type(alphadatadb), parameter **twu31** = alphadatadb(eosid="PR", cid="CL2", ref="tcPR", coeff=(/7.08200000e-01, 8.81200000e-01, 6.30600000e-01/))
- type(cidatadb), parameter **c31** = cidatadb(eosid="PR", cid="CL2", ref="tcPR", bib_ref="Le Guennec et al. (2016): 10.1016/j.fluid.2016.09.003", ciA=-1.48230000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu32** = alphadatadb(eosid="SRK", cid="CL2", ref="tcRK", coeff=(/3.92300000e-01, 8.52400000e-01, 1.21020000e+00/))
- type(cidatadb), parameter **c32** = cidatadb(eosid="SRK", cid="CL2", ref="tcRK", bib_ref="Le Guennec et al. (2016): 10.1016/j.fluid.2016.09.003", ciA=5.77920000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)

- type(gendatadb), parameter **cx18** = gendatadb(ident = "ClF3Si", formula = "ClF3Si", name = "CHLOROTRIFLUOROSILANE", mw = 120.5000, Tc = 307.7000, Pc = 3470000.00, Zc = 0.000000, acf = 0.270730, Tb = 203.2000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = -1.000000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp19** = cpdata(cid = "ClF3Si", ref = "Default", bib_ref = "", cptype = 4, cp = (/9.59700000e+01,-6.08380000e+00,1.20890000e-02,-6.90760000e-06,0.00000000e+00, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(gendatadb), parameter **cx19** = gendatadb(ident = "CYCLOHEX", formula = "C6H12", name = "CYCLOHEXANE", mw = 84.1610, Tc = 553.5000, Pc = 4073000.00, Zc = 0.273000, acf = 0.211000, Tb = 353.9300, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.273000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp20** = cpdata(cid = "CYCLOHEX", ref = "Default", bib_ref = "", cptype = 8, cp = (/4.03500000e+00,-4.43300000e-03,1.68340000e-04,-2.07750000e-07,7.74600000e-11, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -100.0000, Tcpmax = 1000.0000)
- type(alphadatadb), parameter **twu33** = alphadatadb(eosid="PR", cid="CYCLOHEX", ref="tcPR", coeff=(/3.69100000e-01, 8.12000000e-01, 1.36170000e+00/))
- type(cidatadb), parameter **c33** = cidatadb(eosid="PR", cid="CYCLOHEX", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=-4.23270000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu34** = alphadatadb(eosid="SRK", cid="CYCLOHEX", ref="tcRK", coeff=(/3.16900000e-01, 8.33200000e-01, 1.86290000e+00/))
- type(cidatadb), parameter **c34** = cidatadb(eosid="SRK", cid="CYCLOHEX", ref="tcRK", bib_ref="10.1016/j.fluid.2016.09.003", ciA=1.33377000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx20** = gendatadb(ident = "C3_1", formula = "C3H6", name = "CYCLOPROPANE", mw = 42.0810, Tc = 397.8000, Pc = 5490000.00, Zc = 0.274000, acf = 0.130000, Tb = 240.3000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.271600, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp21** = cpdata(cid = "C3_1", ref = "Default", bib_ref = "", cptype = 4, cp = (/-3.52400000e+01,3.81300000e-01,-2.88100000e-04,9.03500000e-08,0.00000000e+00, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu35** = alphadatadb(eosid="PR", cid="C3_1", ref="tcPR", coeff=(/2.16300000e-01, 8.56300000e-01, 1.73750000e+00/))
- type(cidatadb), parameter **c35** = cidatadb(eosid="PR", cid="C3_1", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=-3.05740000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu36** = alphadatadb(eosid="SRK", cid="C3_1", ref="tcRK", coeff=(/2.28700000e-01, 8.71400000e-01, 2.15940000e+00/))
- type(cidatadb), parameter **c36** = cidatadb(eosid="SRK", cid="C3_1", ref="tcRK", bib_ref="10.1016/j.fluid.2016.09.003", ciA=6.36470000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx21** = gendatadb(ident = "D2", formula = "D2", name = "DEUTERIUM", mw = 4.0282, Tc = 38.3400, Pc = 1679600.00, Zc = 0.304200, acf = -0.136290, Tb = 23.6610, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.315000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp22** = cpdata(cid = "D2", ref = "Default", bib_ref = "", cptype = 4, cp = (/2.07860000e+01,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -175.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu37** = alphadatadb(eosid="PR", cid="D2", ref="tcPR", coeff=(/1.48600000e-01, 9.96800000e-01, 1.05870000e+00/))
- type(alphadatadb), parameter **twu38** = alphadatadb(eosid="PR", cid="D2", ref="QuantumCubic", coeff=(/5.50070000e+01, -1.69810000e-02, 3.16210000e+00/))

- type(cidatadb), parameter **c37** = cidatadb(eosid="PR", cid="D2", ref="tcPR", bib_ref="10.1016/j.fluid.2016.↩ 09.003", ciA=-4.55550000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cidatadb), parameter **c38** = cidatadb(eosid="PR", cid="D2", ref="QuantumCubic", bib_ref="10.1016/j.↩ fluid.2020.112790", ciA=-3.87180000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu39** = alphadatadb(eosid="SRK", cid="D2", ref="tcRK", coeff=(/2.↩ 15000000e-01, 9.92100000e-01, 1.10790000e+00/))
- type(cidatadb), parameter **c39** = cidatadb(eosid="SRK", cid="D2", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-1.25430000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx22** = gendatadb(ident = "S434", formula = "C12H26O", name = "DI-n-HEXYL ETHER", mw = 186.3390, Tc = 657.0000, Pc = 1823900.00, Zc = 0.240000, acf = 0.700000, Tb = 499.6000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.63372000e+01, 3.98278000e+03, -8.91500000e+01/), Tantmin = 373.0000, Tantmax = 545.0000, Zra = -1.000000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp23** = cpdata(cid = "S434", ref = "Default", bib_ref = "", cptype = 1, cp = (/8.↩ 01000000e+00,2.56400000e-01,-1.32200000e-04,4.00700000e-08,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(gendatadb), parameter **cx23** = gendatadb(ident = "DME", formula = "C2H6O", name = "di-methyl ether", mw = 46.0684, Tc = 400.3780, Pc = 5336800.00, Zc = 0.269890, acf = 0.196000, Tb = 248.3680, Ttr = 131.6600, Ptr = 2.2000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.↩ 00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = -1.000000, mu_dipole = 1.300000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp24** = cpdata(cid = "DME", ref = "Default", bib_ref = "The properties of gases and liquids, 5th ed. ISBN: 978-0-07-011682-5", cptype = 8, cp = (/4.36100000e+00,6.07000000e-03,2.↩ 89900000e-05,-3.58100000e-08,1.28200000e-11, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00/), Tcpmin = 100.0000, Tcpmax = 1000.0000)
- type(gendatadb), parameter **cx24** = gendatadb(ident = "N2O4", formula = "N2O4", name = "DINITROGEN TETROXIDE", mw = 92.0110, Tc = 431.0100, Pc = 10100000.00, Zc = 0.470700, acf = 1.007000, Tb = 302.↩ 2200, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.70046000e+01, 2.73020000e+03, -3.89700000e+01/), Tantmin = 254.1700, Tantmax = 320.6900, Zra = 0.366500, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp25** = cpdata(cid = "N2O4", ref = "Default", bib_ref = "", cptype = 6, cp = (/3.↩ 04890000e-01,2.46305000e-03,-1.73230000e-06,-5.56640000e-10,7.76230000e-13, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 50.0000, Tcpmax = 1000.↩ 0000)
- type(gendatadb), parameter **cx25** = gendatadb(ident = "E-H2", formula = "H2", name = "EQUILIBRIUM-HYDROGEN", mw = 2.0159, Tc = 32.9380, Pc = 1285800.00, Zc = 0.302000, acf = -0.219000, Tb = 20.2710, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 10, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 15.0000, Tantmax = 32.5000, Zra = 0.306000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp26** = cpdata(cid = "E-H2", ref = "Default", bib_ref = "", cptype = 10, cp = (/2.86719970e+01,1.33961560e+01,2.96013100e-03,-3.98074400e-06,2.66166700e-09, -6.09986300e-13,-1.18013710e+01,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 5.0000, Tcpmax = 5.0000)
- type(gendatadb), parameter **cx26** = gendatadb(ident = "C2", formula = "C2H6", name = "ETHANE", mw = 30.0700, Tc = 305.4000, Pc = 4883900.00, Zc = 0.285000, acf = 0.098000, Tb = 184.5000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.56637000e+01, 1.51142000e+03, -1.71600000e+01/), Tantmin = 130.0000, Tantmax = 199.0000, Zra = 0.280800, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp27** = cpdata(cid = "C2", ref = "Default", bib_ref = "", cptype = 2, cp = (/-4.↩ 93340000e-02,1.10899200e+00,-1.88512000e-04,3.96558000e-06,-3.14020900e-09, 8.00818700e-13,1.↩ 99588900e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -175.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu40** = alphadatadb(eosid="PR", cid="C2", ref="tcPR", coeff=(/1.45900000e-01, 8.78000000e-01, 2.15360000e+00/))
- type(alphadatadb), parameter **mc10** = alphadatadb(eosid="PR", cid="C2", ref="Default", coeff=(/7.↩ 17800000e-01, -7.64400000e-01, 1.63960000e+00/))

- type(alphadatadb), parameter **mc11** = alphadatadb(eosid="PR", cid="C2", ref="Chapoy2005", coeff=(/5.↩ 31000000e-01, -6.20000000e-02, 2.14000000e-01/))
- type(cidatadb), parameter **c40** = cidatadb(eosid="PR", cid="C2", ref="tcPR", bib_ref="10.1016/j.fluid.2016.↩ 09.003", ciA=-3.70740000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu41** = alphadatadb(eosid="SRK", cid="C2", ref="tcRK", coeff=(/2.↩ 24000000e-01, 8.81600000e-01, 2.10900000e+00/))
- type(alphadatadb), parameter **mc12** = alphadatadb(eosid="SRK", cid="C2", ref="Default", coeff=(/7.↩ 17800000e-01, -7.64400000e-01, 1.63960000e+00/))
- type(alphadatadb), parameter **mc13** = alphadatadb(eosid="SRK", cid="C2", ref="Chapoy2005", coeff=(/7.↩ 11000000e-01, -5.73000000e-01, 8.94000000e-01/))
- type(cidatadb), parameter **c41** = cidatadb(eosid="SRK", cid="C2", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=4.57390000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx27** = gendatadb(ident = "ETOH", formula = "ETOH", name = "ETHANOL", mw = 46.0684, Tc = 514.7100, Pc = 6268000.00, Zc = 0.247000, acf = 0.646000, Tb = 351.5700, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.243000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp28** = cpdata(cid = "ETOH", ref = "Default", bib_ref = "", cptype = 2, cp = (/0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00, 0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -175.↩ 0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu42** = alphadatadb(eosid="PR", cid="ETOH", ref="tcPR", coeff=(/9.↩ 86100000e-01, 9.64200000e-01, 1.33380000e+00/))
- type(cidatadb), parameter **c42** = cidatadb(eosid="PR", cid="ETOH", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=5.95100000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu43** = alphadatadb(eosid="SRK", cid="ETOH", ref="tcRK", coeff=(/9.↩ 43700000e-01, 9.36300000e-01, 1.55900000e+00/))
- type(cidatadb), parameter **c43** = cidatadb(eosid="SRK", cid="ETOH", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.60736000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cpadata), parameter **cpa9** = CPAdata(eosid="CPA-SRK", compName="ETOH", ref="SINTEF/Queimada2005", bib_reference="10.1016/j.fluid.2004.08.011", a0=8.67160000e+05, b=4.91100000e-02, eps=2.15320000e+04, beta=8.00000000e-03, alphacorridx = cbAlphaClassicIdx, alphaParams = (/7.36900000e-01,0.↩ 00000000e+00,0.00000000e+00/), assoc_scheme = assoc_scheme_2B)
- type(cpadata), parameter **cpa10** = CPAdata(eosid="CPA-SRK", compName="ETOH", ref="Default/Oliveira2008", bib_reference="10.1016/j.fluid.2008.02.020", a0=6.84150000e+05, b=4.75080000e-02, eps=2.13360000e+04, beta=1.92120000e-02, alphacorridx = cbAlphaClassicIdx, alphaParams = (/9.39230000e-01,0.↩ 00000000e+00,0.00000000e+00/), assoc_scheme = assoc_scheme_2B)
- type(gendatadb), parameter **cx28** = gendatadb(ident = "EBZN", formula = "C8H10", name = "ETHYLBENZENE", mw = 106.1670, Tc = 617.1600, Pc = 3608000.00, Zc = 0.263000, acf = 0.302000, Tb = 409.3600, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.↩ 00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.262000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp29** = cpdata(cid = "EBZN", ref = "Default", bib_ref = "", cptype = 7, cp = (/7.84400000e+04,3.39900000e+05,1.55900000e+03,2.42600000e+05,-7.02000000e+02, 0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 200.↩ 0000, Tcpmax = 1500.0000)
- type(alphadatadb), parameter **twu44** = alphadatadb(eosid="PR", cid="EBZN", ref="tcPR", coeff=(/4.↩ 99100000e-01, 8.22900000e-01, 1.35530000e+00/))
- type(cidatadb), parameter **c44** = cidatadb(eosid="PR", cid="EBZN", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=2.70300000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu45** = alphadatadb(eosid="SRK", cid="EBZN", ref="tcRK", coeff=(/5.↩ 11300000e-01, 8.33400000e-01, 1.56830000e+00/))
- type(cidatadb), parameter **c45** = cidatadb(eosid="SRK", cid="EBZN", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=2.46150000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx29** = gendatadb(ident = "C2_1", formula = "C2H4", name = "ETHYLENE", mw = 28.0540, Tc = 282.4000, Pc = 5035900.00, Zc = 0.276000, acf = 0.085000, Tb = 169.4000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.55368000e+01,

1.34701000e+03, -1.81500000e+01/), Tantmin = 120.0000, Tantmax = 182.0000, Zra = 0.281500, mu_dipole = 0.000000, q_quadrupole = 2.000000)

- type(cpdata), parameter **cp30** = cpdata(cid = "C2_1", ref = "Default", bib_ref = "", cptype = 2, cp = (/6.00935360e+01,6.06930000e-01,1.28878800e-03,1.03363600e-06,-1.09953700e-09, 2.92932600e-13,4.48985300e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -20.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **mc14** = alphadatadb(eosid="PR", cid="C2_1", ref="Chapoy2005", coeff=(/5.↩ 12000000e-01, -8.70000000e-02, 3.49000000e-01/))
- type(alphadatadb), parameter **mc15** = alphadatadb(eosid="SRK", cid="C2_1", ref="Chapoy2005", coeff=(/6.52000000e-01, -3.15000000e-01, 5.63000000e-01/))
- type(gendatadb), parameter **cx30** = gendatadb(ident = "HE", formula = "HE", name = "HELIUM-4", mw = 4.0030, Tc = 5.1953, Pc = 227600.00, Zc = 0.301000, acf = -0.385000, Tb = 4.2100, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.22514000e+01, 3.37329000e+01, 1.79000000e+00/), Tantmin = 3.7000, Tantmax = 4.3000, Zra = 0.335500, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp31** = cpdata(cid = "HE", ref = "Default", bib_ref = "", cptype = 4, cp = (/2.07860000e+01,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00, 0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu46** = alphadatadb(eosid="PR", cid="HE", ref="tcPR", coeff=(/6.30000000e-03, 1.21750000e+00, 1.09090000e+00/))
- type(alphadatadb), parameter **twu47** = alphadatadb(eosid="PR", cid="HE", ref="QuantumCubic", coeff=(/4.85580000e-01, 1.71730000e+00, 3.02710000e-01/))
- type(cidatadb), parameter **c46** = cidatadb(eosid="PR", cid="HE", ref="tcPR", bib_ref="10.1016/j.fluid.2016.↩ 09.003", ciA=-4.89150000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cidatadb), parameter **c47** = cidatadb(eosid="PR", cid="HE", ref="QuantumCubic", bib_ref="10.1016/j.↩ fluid.2020.112790", ciA=-3.17910000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu48** = alphadatadb(eosid="SRK", cid="HE", ref="tcRK", coeff=(/-4.↩ 66000000e-02, 1.24730000e+00, 5.40100000e-01/))
- type(cidatadb), parameter **c48** = cidatadb(eosid="SRK", cid="HE", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-1.46080000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx31** = gendatadb(ident = "N2H4", formula = "N2H4", name = "HYDRAZINE", mw = 32.0452, Tc = 653.1500, Pc = 14700000.00, Zc = 0.280000, acf = 0.314300, Tb = 386.6500, Ttr = 0.0000, Ptr = 0.0000, sref = 238.6600, href = 95353.4000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.264100, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp32** = cpdata(cid = "N2H4", ref = "Default", bib_ref = "Poling, Prausnitz and O'Connell. ISBN: 978-0-07-011682-5", cptype = 8, cp = (/3.62700000e+00,2.23900000e-03,2.↩ 87600000e-05,-4.06000000e-08,1.69000000e-11, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00/), Tcpmin = 50.0000, Tcpmax = 1000.0000)
- type(alphadatadb), parameter **twu49** = alphadatadb(eosid="PR", cid="N2H4", ref="tcPR", coeff=(/4.↩ 60900000e-01, 8.42800000e-01, 1.52790000e+00/))
- type(cidatadb), parameter **c49** = cidatadb(eosid="PR", cid="N2H4", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.13000000e-08, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu50** = alphadatadb(eosid="SRK", cid="N2H4", ref="tcRK", coeff=(/4.↩ 30600000e-01, 8.48400000e-01, 1.89450000e+00/))
- type(cidatadb), parameter **c50** = cidatadb(eosid="SRK", cid="N2H4", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=5.66000000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx32** = gendatadb(ident = "H2", formula = "H2", name = "HYDROGEN", mw = 2.0160, Tc = 33.1450, Pc = 1296400.00, Zc = 0.305000, acf = -0.220000, Tb = 20.4000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.36333000e+01, 1.64900000e+02, 3.19000000e+00/), Tantmin = 14.0000, Tantmax = 25.0000, Zra = 0.306000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp33** = cpdata(cid = "H2", ref = "Default", bib_ref = "", cptype = 2, cp = (/2.86719970e+01,1.33961560e+01,2.96013100e-03,-3.98074400e-06,2.66166700e-09, -6.09986300e-13,-1.18013710e+01,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -175.0000, Tcpmax = 1200.0000)

- type(alphadatadb), parameter **twu51** = alphadatadb(eosid="PR", cid="H2", ref="tcPR", coeff=(/1.↩ 51470000e+00, -3.79590000e+00, -1.37700000e-01/))
- type(alphadatadb), parameter **twu52** = alphadatadb(eosid="PR", cid="H2", ref="QuantumCubic", coeff=(/1.↩ 56210000e+02, -6.20720000e-03, 5.04700000e+00/))
- type(alphadatadb), parameter **mc16** = alphadatadb(eosid="PR", cid="H2", ref="Chapoy2005", coeff=(/9.↩ 50000000e-02, -2.75000000e-01, -2.90000000e-02/))
- type(cidatadb), parameter **c51** = cidatadb(eosid="PR", cid="H2", ref="tcPR", bib_ref="10.1016/j.fluid.2016.↩ 09.003", ciA=-5.33860000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cidatadb), parameter **c52** = cidatadb(eosid="PR", cid="H2", ref="QuantumCubic", bib_ref="10.1016/j.↩ fluid.2020.112790", ciA=-3.81390000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu53** = alphadatadb(eosid="SRK", cid="H2", ref="tcRK", coeff=(/9.↩ 44400000e-01, 3.00870000e+00, 1.76200000e-01/))
- type(alphadatadb), parameter **mc17** = alphadatadb(eosid="SRK", cid="H2", ref="Chapoy2005", coeff=(/1.↩ 61000000e-01, -2.25000000e-01, -2.32000000e-01/))
- type(cidatadb), parameter **c53** = cidatadb(eosid="SRK", cid="H2", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-1.34490000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx33** = gendatadb(ident = "H2O2", formula = "H2O2", name = "HYDROGEN PEROXIDE", mw = 34.0147, Tc = 730.1500, Pc = 21700000.00, Zc = 0.274800, acf = 0.358200, Tb = 424.↩ 5500, Ttr = 272.7403, Ptr = 0.0000, sref = 232.9500, href = -136106.4000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.267000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp34** = cpdata(cid = "H2O2", ref = "Default", bib_ref = "https://webbook.nist.gov", cptype = 12, cp = (/3.42566700e+01,5.51844500e+01,-3.51544300e+01,9.08744000e+00,-4.22157000e-01, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 2000.0000)
- type(alphadatadb), parameter **twu54** = alphadatadb(eosid="PR", cid="H2O2", ref="tcPR", coeff=(/3.↩ 19100000e-01, 8.64900000e-01, 2.28830000e+00/))
- type(cidatadb), parameter **c54** = cidatadb(eosid="PR", cid="H2O2", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-8.74100000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cidatadb), parameter **c55** = cidatadb(eosid="PR", cid="H2O2", ref="tcPR-ENGINEERING", bib_ref="", ciA=1.79175791e-06, ciB=-4.92958300e-09, ciC=0.00000000e+00, c_type=2)
- type(alphadatadb), parameter **twu55** = alphadatadb(eosid="SRK", cid="H2O2", ref="tcRK", coeff=(/4.↩ 35100000e-01, 8.77500000e-01, 2.16040000e+00/))
- type(cidatadb), parameter **c56** = cidatadb(eosid="SRK", cid="H2O2", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=3.35320000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx34** = gendatadb(ident = "H2S", formula = "H2S", name = "HYDROGEN SULFIDE", mw = 34.0800, Tc = 373.2000, Pc = 8936900.00, Zc = 0.284000, acf = 0.100000, Tb = 212.8000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.61040000e+01, 1.76869000e+03, -2.60600000e+01/), Tantmin = 190.0000, Tantmax = 230.0000, Zra = 0.285500, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp35** = cpdata(cid = "H2S", ref = "Default", bib_ref = "", cptype = 2, cp = (/-1.43704900e+00,9.98865000e-01,-1.84315000e-04,5.57087000e-07,-7.86320000e-11, 6.98500000e-15,1.80540900e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -175.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu56** = alphadatadb(eosid="PR", cid="H2S", ref="tcPR", coeff=(/1.↩ 12000000e-01, 8.68800000e-01, 2.27350000e+00/))
- type(alphadatadb), parameter **mc18** = alphadatadb(eosid="PR", cid="H2S", ref="Chapoy2005", coeff=(/5.↩ 07000000e-01, 8.00000000e-03, 3.42000000e-01/))
- type(cidatadb), parameter **c57** = cidatadb(eosid="PR", cid="H2S", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-2.50560000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu57** = alphadatadb(eosid="SRK", cid="H2S", ref="tcRK", coeff=(/1.↩ 74900000e-01, 8.68600000e-01, 2.27610000e+00/))
- type(alphadatadb), parameter **mc19** = alphadatadb(eosid="SRK", cid="H2S", ref="Chapoy2005", coeff=(/6.↩ 41000000e-01, -1.83000000e-01, 5.13000000e-01/))
- type(cidatadb), parameter **c58** = cidatadb(eosid="SRK", cid="H2S", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=3.01750000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)

- type(gendatadb), parameter **cx35** = gendatadb(ident = "IC4", formula = "C4H10", name = "ISOBUTANE", mw = 58.1240, Tc = 408.1000, Pc = 3647700.00, Zc = 0.283000, acf = 0.176000, Tb = 261.3000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.55381000e+01, 2.03273000e+03, -3.31500000e+01/), Tantmin = 187.0000, Tantmax = 280.0000, Zra = 0.275400, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp36** = cpdata(cid = "IC4", ref = "Default", bib_ref = "", cptype = 2, cp = (/2.↩ 67442080e+01,1.95448000e-01,2.52314300e-03,1.95651000e-07,-7.72615000e-10, 2.38608700e-13,3.↩ 46659500e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -75.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **mc20** = alphadatadb(eosid="PR", cid="IC4", ref="Default", coeff=(/8.↩ 28800000e-01, -8.28500000e-01, 2.32010000e+00/))
- type(alphadatadb), parameter **mc21** = alphadatadb(eosid="PR", cid="IC4", ref="Chapoy2005", coeff=(/6.↩ 52000000e-01, -1.49000000e-01, 5.99000000e-01/))
- type(alphadatadb), parameter **twu58** = alphadatadb(eosid="PR", cid="IC4", ref="tcPR", coeff=(/1.↩ 57500000e-01, 8.60100000e-01, 2.39510000e+00/))
- type(cidatadb), parameter **c59** = cidatadb(eosid="PR", cid="IC4", ref="tcPR", bib_ref="10.1016/j.fluid.2016.↩ 09.003", ciA=-4.07050000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **mc22** = alphadatadb(eosid="SRK", cid="IC4", ref="Default", coeff=(/8.↩ 28800000e-01, -8.28500000e-01, 2.32010000e+00/))
- type(alphadatadb), parameter **mc23** = alphadatadb(eosid="SRK", cid="IC4", ref="Chapoy2005", coeff=(/8.↩ 07000000e-01, -4.32000000e-01, 9.10000000e-01/))
- type(alphadatadb), parameter **twu59** = alphadatadb(eosid="SRK", cid="IC4", ref="tcRK", coeff=(/2.↩ 31300000e-01, 8.62500000e-01, 2.35980000e+00/))
- type(cidatadb), parameter **c60** = cidatadb(eosid="SRK", cid="IC4", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.04875000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx36** = gendatadb(ident = "IC5", formula = "C5H12", name = "ISOPENTANE", mw = 72.1510, Tc = 460.4000, Pc = 3384300.00, Zc = 0.271000, acf = 0.227000, Tb = 301.0000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.↩ 56338000e+01, 2.34867000e+03, -4.00500000e+01/), Tantmin = 216.0000, Tantmax = 322.0000, Zra = 0.↩ 271700, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp37** = cpdata(cid = "IC5", ref = "Default", bib_ref = "", cptype = 2, cp = (/6.42520750e+01,-1.31900000e-01,3.54115600e-03,-1.33322500e-06,2.51463000e-10, -1.29589000e-14,4.57297600e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -20.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **mc24** = alphadatadb(eosid="PR", cid="IC5", ref="Default", coeff=(/8.↩ 76700000e-01, -6.04300000e-01, 1.40250000e+00/))
- type(alphadatadb), parameter **mc25** = alphadatadb(eosid="PR", cid="IC5", ref="Chapoy2005", coeff=(/7.↩ 24000000e-01, -1.66000000e-01, 5.15000000e-01/))
- type(alphadatadb), parameter **twu60** = alphadatadb(eosid="PR", cid="IC5", ref="tcPR", coeff=(/2.↩ 08400000e-01, 8.41800000e-01, 2.13820000e+00/))
- type(cidatadb), parameter **c61** = cidatadb(eosid="PR", cid="IC5", ref="tcPR", bib_ref="10.1016/j.fluid.2016.↩ 09.003", ciA=-3.62110000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **mc26** = alphadatadb(eosid="SRK", cid="IC5", ref="Default", coeff=(/8.↩ 76700000e-01, -6.04300000e-01, 1.40250000e+00/))
- type(alphadatadb), parameter **mc27** = alphadatadb(eosid="SRK", cid="IC5", ref="Chapoy2005", coeff=(/8.↩ 76000000e-01, -3.86000000e-01, 6.60000000e-01/))
- type(alphadatadb), parameter **twu61** = alphadatadb(eosid="SRK", cid="IC5", ref="tcRK", coeff=(/2.↩ 37400000e-01, 8.54800000e-01, 2.47360000e+00/))
- type(cidatadb), parameter **c62** = cidatadb(eosid="SRK", cid="IC5", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.38881000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx37** = gendatadb(ident = "KR", formula = "KR", name = "KRYPTON", mw = 83.7980, Tc = 209.4800, Pc = 5525000.00, Zc = 0.291000, acf = -0.004000, Tb = 119.9300, Ttr = 115.↩ 7750, Ptr = 73530.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.52330000e+01, 7.00510000e+02, -5.84000000e+00/), Tantmin = 81.0000, Tantmax = 94.0000, Zra = 0.308500, mu_dipole = 0.000000, q_quadrupole = 0.000000)

- type(cpdata), parameter **cp38** = cpdata(cid = "KR", ref = "Default", bib_ref = "NIST-Chase1998", cptype = 5, cp = (/2.07860300e+01,4.85063800e-10,-1.58291600e-10,1.52510200e-11,3.19634700e-11, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(gendatadb), parameter **cx38** = gendatadb(ident = "LJF", formula = "LJF", name = "LENNARD-JONES_FLUID", mw = 1.0000, Tc = 132.0000, Pc = 6650000.00, Zc = 0.310000, acf = 0.317700, Tb = 0.8000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = -1.000000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp39** = cpdata(cid = "LJF", ref = "Default", bib_ref = "", cptype = 8, cp = (/2.50000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 1500.0000)
- type(gendatadb), parameter **cx39** = gendatadb(ident = "MXYL", formula = "C8H10", name = "M-XYLENE", mw = 106.1670, Tc = 617.0500, Pc = 3536000.00, Zc = 0.259000, acf = 0.326000, Tb = 412.2700, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.258700, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp40** = cpdata(cid = "MXYL", ref = "Default", bib_ref = "", cptype = 7, cp = (/7.56800000e+04,3.39240000e+05,1.49600000e+03,2.24700000e+05,-6.75900000e+02, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 200.0000, Tcpmax = 1500.0000)
- type(alphadatadb), parameter **twu62** = alphadatadb(eosid="PR", cid="MXYL", ref="tcPR", coeff=(/3.50600000e-01, 8.33200000e-01, 1.85940000e+00/))
- type(cidatadb), parameter **c63** = cidatadb(eosid="PR", cid="MXYL", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=5.14480000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu63** = alphadatadb(eosid="SRK", cid="MXYL", ref="tcRK", coeff=(/3.49200000e-01, 8.48100000e-01, 2.26170000e+00/))
- type(cidatadb), parameter **c64** = cidatadb(eosid="SRK", cid="MXYL", ref="tcRK", bib_ref="10.1016/j.fluid.2016.09.003", ciA=2.70949000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx40** = gendatadb(ident = "C1", formula = "CH4", name = "METHANE", mw = 16.0425, Tc = 190.5550, Pc = 4598837.00, Zc = 0.283742, acf = 0.011310, Tb = 111.7000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.52243000e+01, 8.97840000e+02, -7.16000000e+00/), Tantmin = 93.0000, Tantmax = 120.0000, Zra = 0.289200, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp41** = cpdata(cid = "C1", ref = "Default", bib_ref = "", cptype = 2, cp = (/-1.62285490e+01,2.39359400e+00,-2.21800700e-03,5.74022000e-06,-3.72790500e-09, 8.54968500e-13,-3.39779000e-01,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -175.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu64** = alphadatadb(eosid="PR", cid="C1", ref="tcPR", coeff=(/1.47100000e-01, 9.07400000e-01, 1.82530000e+00/))
- type(alphadatadb), parameter **mc28** = alphadatadb(eosid="PR", cid="C1", ref="Default", coeff=(/5.85700000e-01, -7.20600000e-01, 1.28990000e+00/))
- type(alphadatadb), parameter **mc29** = alphadatadb(eosid="PR", cid="C1", ref="Chapoy2005", coeff=(/4.16000000e-01, -1.73000000e-01, 3.48000000e-01/))
- type(cidatadb), parameter **c65** = cidatadb(eosid="PR", cid="C1", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=-3.56060000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu65** = alphadatadb(eosid="SRK", cid="C1", ref="tcRK", coeff=(/2.17100000e-01, 9.08200000e-01, 1.81720000e+00/))
- type(alphadatadb), parameter **mc30** = alphadatadb(eosid="SRK", cid="C1", ref="Default", coeff=(/5.85700000e-01, -7.20600000e-01, 1.28990000e+00/))
- type(alphadatadb), parameter **mc31** = alphadatadb(eosid="SRK", cid="C1", ref="Chapoy2005", coeff=(/5.49000000e-01, -4.09000000e-01, 6.03000000e-01/))
- type(cidatadb), parameter **c66** = cidatadb(eosid="SRK", cid="C1", ref="tcRK", bib_ref="10.1016/j.fluid.2016.09.003", ciA=2.05030000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)

- type(gendatadb), parameter **cx41** = gendatadb(ident = "MEOH", formula = "CH4O", name = "METHANOL", mw = 32.0420, Tc = 512.6000, Pc = 8095900.00, Zc = 0.224000, acf = 0.559000, Tb = 337.8000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.↩ 85875000e+01, 3.62655000e+03, -3.42900000e+01/), Tantmin = 257.0000, Tantmax = 364.0000, Zra = 0.↩ 233400, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp42** = cpdata(cid = "MEOH", ref = "Default", bib_ref = "", cptype = 1, cp = (/5.↩ 05200000e+00,1.69400000e-02,6.17900000e-06,-6.81100000e-09,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu66** = alphadatadb(eosid="PR", cid="MEOH", ref="tcPR", coeff=(/6.↩ 75500000e-01, 9.14100000e-01, 1.75860000e+00/))
- type(cidatadb), parameter **c67** = cidatadb(eosid="PR", cid="MEOH", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=9.18650000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu67** = alphadatadb(eosid="SRK", cid="MEOH", ref="tcRK", coeff=(/7.↩ 08200000e-01, 9.02200000e-01, 1.87800000e+00/))
- type(cidatadb), parameter **c68** = cidatadb(eosid="SRK", cid="MEOH", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.69543000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cpadata), parameter **cpa11** = CPAdata(eosid="CPA-SRK", compName="MEOH", ref="Default/Kontogeorgis2008", bib_reference="10.2516/ogst:2008025", a0=4.05310000e+05, b=3.09780000e-02, eps=2.45910000e+04, beta=1.61000000e-02, alphacorridx = cbAlphaClassicIdx, alphaParams = (/4.31020000e-01,0.↩ 00000000e+00,0.00000000e+00/), assoc_scheme = assoc_scheme_2B)
- type(gendatadb), parameter **cx42** = gendatadb(ident = "MTC5", formula = "C6H12", name = "METHYLCYCLOPENTANE", mw = 84.1620, Tc = 532.7000, Pc = 3789600.00, Zc = 0.273000, acf = 0.239000, Tb = 345.0000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.58023000e+01, 2.73100000e+03, -4.71100000e+01/), Tantmin = 250.0000, Tantmax = 375.0000, Zra = 0.271100, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp43** = cpdata(cid = "MTC5", ref = "Default", bib_ref = "", cptype = 1, cp = (/-1.↩ 19680000e+01,1.52400000e-01,-8.69900000e-05,1.91400000e-08,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu68** = alphadatadb(eosid="PR", cid="MTC5", ref="tcPR", coeff=(/3.↩ 83900000e-01, 8.08200000e-01, 1.37410000e+00/))
- type(cidatadb), parameter **c69** = cidatadb(eosid="PR", cid="MTC5", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-3.43780000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu69** = alphadatadb(eosid="SRK", cid="MTC5", ref="tcRK", coeff=(/3.↩ 05400000e-01, 8.29300000e-01, 1.96110000e+00/))
- type(cidatadb), parameter **c70** = cidatadb(eosid="SRK", cid="MTC5", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.46862000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx43** = gendatadb(ident = "MEG", formula = "C2H6O2", name = "ETHYLENE GLYCOL", mw = 62.0700, Tc = 720.0000, Pc = 8200000.00, Zc = 0.261600, acf = 0.534700, Tb = 470.2500, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.89090800e+01, 4.97797500e+03, -6.47210000e+01/), Tantmin = 200.0000, Tantmax = 720.0000, Zra = 0.242400, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp44** = cpdata(cid = "MEG", ref = "Default", bib_ref = "", cptype = 2, cp = (/5.↩ 44670000e-01,1.18540000e-03,0.00000000e+00,0.00000000e+00,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -22.0000, Tcpmax = 150.↩ 0000)
- type(alphadatadb), parameter **twu70** = alphadatadb(eosid="PR", cid="MEG", ref="tcPR", coeff=(/1.↩ 57530000e+00, 1.00000000e+00, 6.61400000e-01/))
- type(cidatadb), parameter **c71** = cidatadb(eosid="PR", cid="MEG", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=8.38700000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu71** = alphadatadb(eosid="SRK", cid="MEG", ref="tcRK", coeff=(/1.↩ 54540000e+00, 1.00000000e+00, 7.62500000e-01/))
- type(cidatadb), parameter **c72** = cidatadb(eosid="SRK", cid="MEG", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.92954000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)

- type(gendatadb), parameter **cx44** = gendatadb(ident = "NE", formula = "NE", name = "NEON", mw = 20.1830, Tc = 44.4000, Pc = 2661630.00, Zc = 0.311000, acf = -0.038450, Tb = 27.1000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/2.56553000e+01, -2.95285000e+02, 4.97548000e+00/), Tantmin = 30.0000, Tantmax = 40.0000, Zra = 0.308500, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp45** = cpdata(cid = "NE", ref = "Default", bib_ref = "", cptype = 4, cp = (/2.07860000e+01,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu72** = alphadatadb(eosid="PR", cid="NE", ref="tcPR", coeff=(/1.88700000e-01, 9.47000000e-01, 1.46980000e+00/))
- type(alphadatadb), parameter **twu73** = alphadatadb(eosid="PR", cid="NE", ref="QuantumCubic", coeff=(/4.04530000e-01, 9.58610000e-01, 8.39600000e-01/))
- type(cidatadb), parameter **c73** = cidatadb(eosid="PR", cid="NE", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=-2.35730000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cidatadb), parameter **c74** = cidatadb(eosid="PR", cid="NE", ref="QuantumCubic", bib_ref="10.1016/j.fluid.2020.112790", ciA=-2.46650000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu74** = alphadatadb(eosid="SRK", cid="NE", ref="tcRK", coeff=(/3.27500000e-01, 9.69900000e-01, 1.28930000e+00/))
- type(cidatadb), parameter **c75** = cidatadb(eosid="SRK", cid="NE", ref="tcRK", bib_ref="10.1016/j.fluid.2016.09.003", ciA=-7.12000000e-08, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx45** = gendatadb(ident = "NO", formula = "NO", name = "NITRIC OXIDE", mw = 30.0061, Tc = 180.0000, Pc = 6480000.00, Zc = 0.251127, acf = 0.582000, Tb = 121.3800, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/2.01315000e+01, 1.57250000e+03, -4.88000000e+00/), Tantmin = 106.9000, Tantmax = 127.5600, Zra = 0.266800, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp46** = cpdata(cid = "NO", ref = "Default", bib_ref = "", cptype = 6, cp = (/8.56500000e-01,-1.44390000e-03,3.90260000e-06,-4.07270000e-09,1.52250000e-12, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 50.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu75** = alphadatadb(eosid="PR", cid="NO", ref="tcPR", coeff=(/8.81500000e-01, 9.55200000e-01, 1.40470000e+00/))
- type(cidatadb), parameter **c76** = cidatadb(eosid="PR", cid="NO", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=-7.54000000e-07, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu76** = alphadatadb(eosid="SRK", cid="NO", ref="tcRK", coeff=(/8.68100000e-01, 9.32000000e-01, 1.59540000e+00/))
- type(cidatadb), parameter **c77** = cidatadb(eosid="SRK", cid="NO", ref="tcRK", bib_ref="10.1016/j.fluid.2016.09.003", ciA=2.63650000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx46** = gendatadb(ident = "N2", formula = "N2", name = "NITROGEN", mw = 28.0130, Tc = 126.1610, Pc = 3394400.00, Zc = 0.290000, acf = 0.040000, Tb = 77.4000, Ttr = 0.0000, Ptr = 0.0000, sref = 191.6100, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.49542000e+01, 5.88720000e+02, -6.60000000e+00/), Tantmin = 54.0000, Tantmax = 90.0000, Zra = 0.290000, mu_dipole = 0.000000, q_quadrupole = 1.430000)
- type(cpdata), parameter **cp47** = cpdata(cid = "N2", ref = "Default", bib_ref = "", cptype = 2, cp = (/-2.17250700e+00,1.06849000e+00,-1.34096000e-04,2.15569000e-07,-7.86320000e-11, 6.98500000e-15,1.80540900e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -175.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu77** = alphadatadb(eosid="PR", cid="N2", ref="tcPR", coeff=(/1.24000000e-01, 8.89700000e-01, 2.01380000e+00/))
- type(alphadatadb), parameter **mc32** = alphadatadb(eosid="PR", cid="N2", ref="Default", coeff=(/4.04606000e-01, 3.91057000e-01, -9.63495000e-01/))
- type(alphadatadb), parameter **mc33** = alphadatadb(eosid="PR", cid="N2", ref="Chapoy2005", coeff=(/4.48000000e-01, -1.57000000e-01, 4.69000000e-01/))
- type(cidatadb), parameter **c78** = cidatadb(eosid="PR", cid="N2", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=-3.64220000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu78** = alphadatadb(eosid="SRK", cid="N2", ref="tcRK", coeff=(/1.90200000e-01, 8.90000000e-01, 2.01060000e+00/))

- type(alphadatadb), parameter **mc34** = alphadatadb(eosid="SRK", cid="N2", ref="Default", coeff=(/5.↩
86700000e-01, -4.45900000e-01, 8.92600000e-01/))
- type(alphadatadb), parameter **mc35** = alphadatadb(eosid="SRK", cid="N2", ref="Chapoy2005", coeff=(/5.↩
84000000e-01, -3.96000000e-01, 7.36000000e-01/))
- type(cidatadb), parameter **c79** = cidatadb(eosid="SRK", cid="N2", ref="tcRK", bib_ref="10.1016/j.fluid.↩
2016.09.003", ciA=1.34700000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx47** = gendatadb(ident = "N2O", formula = "N2O", name = "NITROUS OXIDE", mw = 44.0130, Tc = 309.6000, Pc = 7240000.00, Zc = 0.274000, acf = 0.165000, Tb = 184.7000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.↩
00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.275800, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp48** = cpdata(cid = "N2O", ref = "Default", bib_ref = "", cptype = 4, cp = (/2.↩
16200000e+01,7.28100000e-02,-5.77800000e-05,1.83000000e-08,0.00000000e+00, 0.00000000e+00,0.↩
00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu79** = alphadatadb(eosid="PR", cid="N2O", ref="tcPR", coeff=(/6.↩
24800000e-01, 7.93300000e-01, 7.97600000e-01/))
- type(cidatadb), parameter **c80** = cidatadb(eosid="PR", cid="N2O", ref="tcPR", bib_ref="10.1016/j.fluid.↩
2016.09.003", ciA=-1.23720000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu80** = alphadatadb(eosid="SRK", cid="N2O", ref="tcRK", coeff=(/3.↩
08500000e-01, 8.13400000e-01, 1.56750000e+00/))
- type(cidatadb), parameter **c81** = cidatadb(eosid="SRK", cid="N2O", ref="tcRK", bib_ref="10.1016/j.fluid.↩
2016.09.003", ciA=4.39740000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx48** = gendatadb(ident = "N-H2", formula = "H2", name = "N-HYDROGEN", mw = 2.0159, Tc = 33.1450, Pc = 1296400.00, Zc = 0.303000, acf = -0.219000, Tb = 20.3690, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 10, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 15.0989, Tantmax = 32.6952, Zra = 0.306000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp49** = cpdata(cid = "N-H2", ref = "Default", bib_ref = "", cptype = 10, cp = (/2.86719970e+01,1.33961560e+01,2.96013100e-03,-3.98074400e-06,2.66166700e-09, -6.09986300e-13,-1.18013710e+01,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 5.0000, Tcpmax = 5.0000)
- type(gendatadb), parameter **cx49** = gendatadb(ident = "O-H2", formula = "H2", name = "ORTHO-HYDROGEN", mw = 2.0159, Tc = 33.2200, Pc = 1310650.00, Zc = 0.307000, acf = -0.218000, Tb = 20.3800, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 10, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 15.0989, Tantmax = 32.6952, Zra = 0.306000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp50** = cpdata(cid = "O-H2", ref = "Default", bib_ref = "", cptype = 10, cp = (/2.86719970e+01,1.33961560e+01,2.96013100e-03,-3.98074400e-06,2.66166700e-09, -6.09986300e-13,-1.18013710e+01,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 5.0000, Tcpmax = 5.0000)
- type(gendatadb), parameter **cx50** = gendatadb(ident = "OXYL", formula = "C8H10", name = "O-XYLENE", mw = 106.1670, Tc = 630.3300, Pc = 3734000.00, Zc = 0.263000, acf = 0.310400, Tb = 417.5800, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.↩
00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.261600, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp51** = cpdata(cid = "OXYL", ref = "Default", bib_ref = "", cptype = 7, cp = (/8.52100000e+04,3.29540000e+05,1.49440000e+03,2.11500000e+05,-6.75800000e+02, 0.↩
00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 200.↩
0000, Tcpmax = 1500.0000)
- type(alphadatadb), parameter **twu81** = alphadatadb(eosid="PR", cid="OXYL", ref="tcPR", coeff=(/3.↩
10800000e-01, 8.46300000e-01, 2.02890000e+00/))
- type(cidatadb), parameter **c82** = cidatadb(eosid="PR", cid="OXYL", ref="tcPR", bib_ref="10.1016/j.fluid.↩
2016.09.003", ciA=2.52880000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu82** = alphadatadb(eosid="SRK", cid="OXYL", ref="tcRK", coeff=(/3.↩
20600000e-01, 8.58600000e-01, 2.41770000e+00/))

- type(cidatadb), parameter **c83** = cidatadb(eosid="SRK", cid="OXYL", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=2.38251000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx51** = gendatadb(ident = "O2", formula = "O2", name = "OXYGEN", mw = 31.↩ 9990, Tc = 154.6000, Pc = 5045990.00, Zc = 0.288000, acf = 0.021000, Tb = 90.2000, Ttr = 0.0000, Ptr = 0.0000, sref = 205.1500, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.54075000e+01, 7.34550000e+02, -6.45000000e+00/), Tantmin = 63.0000, Tantmax = 100.0000, Zra = 0.290500, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp52** = cpdata(cid = "O2", ref = "Default", bib_ref = "", cptype = 2, cp = (/-2.↩ 28357400e+00,9.52440000e-01,-2.81140000e-04,6.55223000e-07,-4.52316000e-10, 1.08774400e-13,2.↩ 08031000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -175.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu83** = alphadatadb(eosid="PR", cid="O2", ref="tcPR", coeff=(/2.12900000e-01, 8.91300000e-01, 1.40050000e+00/))
- type(alphadatadb), parameter **mc36** = alphadatadb(eosid="PR", cid="O2", ref="Chapoy2005", coeff=(/4.↩ 13000000e-01, -1.70000000e-02, 9.20000000e-02/))
- type(cidatadb), parameter **c84** = cidatadb(eosid="PR", cid="O2", ref="tcPR", bib_ref="10.1016/j.fluid.2016.↩ 09.003", ciA=-2.76670000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu84** = alphadatadb(eosid="SRK", cid="O2", ref="tcRK", coeff=(/2.↩ 11800000e-01, 9.02000000e-01, 1.87980000e+00/))
- type(alphadatadb), parameter **mc37** = alphadatadb(eosid="SRK", cid="O2", ref="Chapoy2005", coeff=(/5.↩ 45000000e-01, -2.35000000e-01, 2.92000000e-01/))
- type(cidatadb), parameter **c85** = cidatadb(eosid="SRK", cid="O2", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.34570000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx52** = gendatadb(ident = "P-H2", formula = "H2", name = "PARA-HYDROGEN", mw = 2.0159, Tc = 32.9380, Pc = 1285800.00, Zc = 0.302000, acf = -0.219000, Tb = 20.2710, Ttr = 0.↩ 0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 10, ant = (/0.↩ 00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 15.0000, Tantmax = 32.5000, Zra = 0.↩ 306000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp53** = cpdata(cid = "P-H2", ref = "Default", bib_ref = "", cptype = 10, cp = (/2.86719970e+01,1.33961560e+01,2.96013100e-03,-3.98074400e-06,2.66166700e-09, -6.09986300e-13,-1.18013710e+01,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 5.0000, Tcpmax = 5.0000)
- type(gendatadb), parameter **cx53** = gendatadb(ident = "PXYL", formula = "C8H10", name = "P-XYLENE", mw = 106.1670, Tc = 616.2300, Pc = 3511000.00, Zc = 0.260000, acf = 0.321500, Tb = 411.5100, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.↩ 00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.258500, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp54** = cpdata(cid = "PXYL", ref = "Default", bib_ref = "", cptype = 7, cp = (/7.51200000e+04,3.39700000e+05,1.49280000e+03,2.24700000e+05,-6.75100000e+02, 0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 200.↩ 0000, Tcpmax = 1500.0000)
- type(alphadatadb), parameter **twu85** = alphadatadb(eosid="PR", cid="PXYL", ref="tcPR", coeff=(/2.↩ 26200000e-01, 8.50100000e-01, 2.54710000e+00/))
- type(cidatadb), parameter **c86** = cidatadb(eosid="PR", cid="PXYL", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=5.37320000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu86** = alphadatadb(eosid="SRK", cid="PXYL", ref="tcRK", coeff=(/2.↩ 97900000e-01, 8.51400000e-01, 2.52780000e+00/))
- type(cidatadb), parameter **c87** = cidatadb(eosid="SRK", cid="PXYL", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=2.76019000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx54** = gendatadb(ident = "C3", formula = "C3H8", name = "PROPANE", mw = 44.0970, Tc = 369.8000, Pc = 4245500.00, Zc = 0.281000, acf = 0.152000, Tb = 231.1000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.57260000e+01, 1.87246000e+03, -2.51600000e+01/), Tantmin = 164.0000, Tantmax = 249.0000, Zra = 0.276600, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp55** = cpdata(cid = "C3", ref = "Default", bib_ref = "", cptype = 7, cp = (/5.19200000e+04,1.92450000e+05,1.62650000e+03,1.16800000e+05,7.23600000e+02, 0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 200.↩ 0000, Tcpmax = 1500.0000)

- type(alphadatadb), parameter **twu87** = alphadatadb(eosid="PR", cid="C3", ref="tcPR", coeff=(/1.59600000e-01, 8.68100000e-01, 2.28200000e+00/))
- type(alphadatadb), parameter **mc38** = alphadatadb(eosid="PR", cid="C3", ref="Default", coeff=(/7.↩ 86300000e-01, -7.45900000e-01, 1.84540000e+00/))
- type(alphadatadb), parameter **mc39** = alphadatadb(eosid="PR", cid="C3", ref="Chapoy2005", coeff=(/6.↩ 00000000e-01, -6.00000000e-03, 1.74000000e-01/))
- type(cidatadb), parameter **c88** = cidatadb(eosid="PR", cid="C3", ref="tcPR", bib_ref="10.1016/j.fluid.2016.↩ 09.003", ciA=-3.89270000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu88** = alphadatadb(eosid="SRK", cid="C3", ref="tcRK", coeff=(/2.↩ 45300000e-01, 8.73700000e-01, 2.20880000e+00/))
- type(alphadatadb), parameter **mc40** = alphadatadb(eosid="SRK", cid="C3", ref="Default", coeff=(/7.↩ 86300000e-01, -7.45900000e-01, 1.84540000e+00/))
- type(alphadatadb), parameter **mc41** = alphadatadb(eosid="SRK", cid="C3", ref="Chapoy2005", coeff=(/7.↩ 75000000e-01, -4.76000000e-01, 8.15000000e-01/))
- type(cidatadb), parameter **c89** = cidatadb(eosid="SRK", cid="C3", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=7.46600000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cpadata), parameter **cpa12** = CPAdata(eosid="CPA-SRK", compName="C3", ref="Default/Kontogeorgis-Folas2010", bib_reference="10.1002/9780470747537", a0=9.11875000e+05, b=5.78340000e-02, eps=0.↩ 00000000e+00, beta=0.00000000e+00, alphacorridx = cbAlphaClassicIdx, alphaParams = (/6.30700000e-01,0.00000000e+00,0.00000000e+00/), assoc_scheme = no_assoc)
- type(gendatadb), parameter **cx55** = gendatadb(ident = "PRLN", formula = "C3H6", name = "PROPYLENE", mw = 42.0810, Tc = 364.9000, Pc = 4600000.00, Zc = 0.274000, acf = 0.144000, Tb = 225.5000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.↩ 57027000e+01, 1.80753000e+03, -2.61500000e+01/), Tantmin = 160.0000, Tantmax = 240.0000, Zra = 0.↩ 277900, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp56** = cpdata(cid = "PRLN", ref = "Default", bib_ref = "", cptype = 2, cp = (/6.63699910e+01,1.28994000e-01,2.64691000e-03,-6.71019000e-07,-5.52250000e-11, 4.94690000e-14,5.11755300e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -20.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu89** = alphadatadb(eosid="PR", cid="PRLN", ref="tcPR", coeff=(/4.↩ 64100000e-01, 8.41900000e-01, 1.04550000e+00/))
- type(cidatadb), parameter **c90** = cidatadb(eosid="PR", cid="PRLN", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-3.56200000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu90** = alphadatadb(eosid="SRK", cid="PRLN", ref="tcRK", coeff=(/3.↩ 84900000e-01, 8.51300000e-01, 1.46570000e+00/))
- type(cidatadb), parameter **c91** = cidatadb(eosid="SRK", cid="PRLN", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=6.89870000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx56** = gendatadb(ident = "R11", formula = "CCL3F", name = "TRICHLOROFLUOROMETHANE", mw = 137.3680, Tc = 471.2000, Pc = 4407600.00, Zc = 0.279000, acf = 0.188000, Tb = 297.0000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.58516000e+01, 2.40161000e+03, -3.63000000e+01/), Tantmin = 240.0000, Tantmax = 300.0000, Zra = 0.274500, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp57** = cpdata(cid = "R11", ref = "Default", bib_ref = "", cptype = 1, cp = (/9.↩ 78900000e+00,3.89300000e-02,-3.38300000e-05,9.90300000e-09,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu91** = alphadatadb(eosid="PR", cid="R11", ref="tcPR", coeff=(/3.↩ 33800000e-01, 8.31800000e-01, 1.44220000e+00/))
- type(cidatadb), parameter **c92** = cidatadb(eosid="PR", cid="R11", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-4.75660000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu92** = alphadatadb(eosid="SRK", cid="R11", ref="tcRK", coeff=(/3.↩ 24500000e-01, 8.48400000e-01, 1.82270000e+00/))
- type(cidatadb), parameter **c93** = cidatadb(eosid="SRK", cid="R11", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=9.15060000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx57** = gendatadb(ident = "R1114", formula = "C2F4", name = "TETRAFLUOROETHYLENE", mw = 100.0160, Tc = 306.5000, Pc = 3940000.00, Zc = 0.267000, acf = 0.223000, Tb = 197.2000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode =

2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.270100, mu_dipole = 0.000000, q_quadrupole = 0.000000)

- type(cpdata), parameter **cp58** = cpdata(cid = "R1114", ref = "Default", bib_ref = "", cptype = 4, cp = (/2.↩ 90100000e+01,2.27700000e-01,-2.03600000e-04,6.77800000e-08,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu93** = alphadatadb(eosid="PR", cid="R1114", ref="tcPR", coeff=(/2.↩ 84800000e-01, 8.17800000e-01, 1.67930000e+00/))
- type(cidatadb), parameter **c94** = cidatadb(eosid="PR", cid="R1114", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-1.39410000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu94** = alphadatadb(eosid="SRK", cid="R1114", ref="tcRK", coeff=(/2.↩ 82900000e-01, 8.36700000e-01, 2.09660000e+00/))
- type(cidatadb), parameter **c95** = cidatadb(eosid="SRK", cid="R1114", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=8.62210000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx58** = gendatadb(ident = "R1132a", formula = "C2H2F2", name = "1,1-DIFLUOROETHYLENE", mw = 64.0350, Tc = 302.9000, Pc = 4460000.00, Zc = 0.273000, acf = 0.140000, Tb = 187.5000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.271300, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp59** = cpdata(cid = "R1132a", ref = "Default", bib_ref = "", cptype = 4, cp = (/3.↩ 07300000e+00,2.44500000e-01,-2.09900000e-04,7.02100000e-08,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu95** = alphadatadb(eosid="PR", cid="R1132a", ref="tcPR", coeff=(/6.↩ 79000000e-02, 8.44700000e-01, 2.63710000e+00/))
- type(cidatadb), parameter **c96** = cidatadb(eosid="PR", cid="R1132a", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-4.08900000e-07, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu96** = alphadatadb(eosid="SRK", cid="R1132a", ref="tcRK", coeff=(/1.↩ 35000000e-01, 8.46200000e-01, 2.61180000e+00/))
- type(cidatadb), parameter **c97** = cidatadb(eosid="SRK", cid="R1132a", ref="tcRK", bib_ref="10.1016/j.↩ fluid.2016.09.003", ciA=8.45540000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx59** = gendatadb(ident = "R114", formula = "C2CL2F4", name = "1,2-DICHLOROTETRAFLUOROETHANE", mw = 170.9220, Tc = 418.9000, Pc = 3262700.00, Zc = 0.275000, acf = 0.255000, Tb = 277.0000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.273700, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp60** = cpdata(cid = "R114", ref = "Default", bib_ref = "", cptype = 1, cp = (/9.↩ 26200000e+00,8.21600000e-02,-7.04700000e-05,2.03200000e-08,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu97** = alphadatadb(eosid="PR", cid="R114", ref="tcPR", coeff=(/1.↩ 49200000e-01, 8.43900000e-01, 2.65050000e+00/))
- type(cidatadb), parameter **c98** = cidatadb(eosid="PR", cid="R114", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-5.03720000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu98** = alphadatadb(eosid="SRK", cid="R114", ref="tcRK", coeff=(/2.↩ 21100000e-01, 8.45900000e-01, 2.61660000e+00/))
- type(cidatadb), parameter **c99** = cidatadb(eosid="SRK", cid="R114", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.14150000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx60** = gendatadb(ident = "R115", formula = "C2CLF5", name = "CHLOROPENTAFLUOROETHANE", mw = 154.4670, Tc = 353.2000, Pc = 3161300.00, Zc = 0.271000, acf = 0.253000, Tb = 234.0000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.57343000e+01, 1.84890000e+03, -3.08800000e+01/), Tantmin = 175.0000, Tantmax = 230.0000, Zra = 0.275700, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp61** = cpdata(cid = "R115", ref = "Default", bib_ref = "", cptype = 1, cp = (/6.↩ 64800000e+00,8.34000000e-02,-6.90400000e-05,1.94400000e-08,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)

- type(alphadatadb), parameter **twu99** = alphadatadb(eosid="PR", cid="R115", ref="tcPR", coeff=(/7.↩ 21200000e-01, 8.70300000e-01, 9.54000000e-01/))
- type(cidatadb), parameter **c100** = cidatadb(eosid="PR", cid="R115", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-6.21940000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu100** = alphadatadb(eosid="SRK", cid="R115", ref="tcRK", coeff=(/3.↩ 45600000e-01, 8.39400000e-01, 1.94490000e+00/))
- type(cidatadb), parameter **c101** = cidatadb(eosid="SRK", cid="R115", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=8.07540000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx61** = gendatadb(ident = "R116", formula = "C2F6", name = "HEXAFLUOROETHANE", mw = 138.0120, Tc = 293.0000, Pc = 3060000.00, Zc = 0.279000, acf = 0.253680, Tb = 194.9000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.277800, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp62** = cpdata(cid = "R116", ref = "Default", bib_ref = "", cptype = 4, cp = (/2.↩ 68200000e+01,3.45800000e-01,-2.86900000e-04,8.13500000e-08,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu101** = alphadatadb(eosid="PR", cid="R116", ref="tcPR", coeff=(/2.↩ 20000000e-01, 8.32600000e-01, 2.13210000e+00/))
- type(cidatadb), parameter **c102** = cidatadb(eosid="PR", cid="R116", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-6.70800000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu102** = alphadatadb(eosid="SRK", cid="R116", ref="tcRK", coeff=(/2.↩ 27400000e-01, 8.46900000e-01, 2.59910000e+00/))
- type(cidatadb), parameter **c103** = cidatadb(eosid="SRK", cid="R116", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=5.60370000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx62** = gendatadb(ident = "R12", formula = "CCL2F2", name = "DICHLORODIFLUOROMETHANE", mw = 120.9140, Tc = 385.0000, Pc = 4123900.00, Zc = 0.280000, acf = 0.176000, Tb = 243.4000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.275700, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp63** = cpdata(cid = "R12", ref = "Default", bib_ref = "", cptype = 1, cp = (/7.↩ 54700000e+00,4.25700000e-02,-3.60300000e-05,1.03700000e-08,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu103** = alphadatadb(eosid="PR", cid="R12", ref="tcPR", coeff=(/1.↩ 55200000e-01, 8.60100000e-01, 2.39040000e+00/))
- type(cidatadb), parameter **c104** = cidatadb(eosid="PR", cid="R12", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-4.23300000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu104** = alphadatadb(eosid="SRK", cid="R12", ref="tcRK", coeff=(/2.↩ 18200000e-01, 8.60000000e-01, 2.39650000e+00/))
- type(cidatadb), parameter **c105** = cidatadb(eosid="SRK", cid="R12", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=7.93010000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx63** = gendatadb(ident = "R1234yf", formula = "CF3CF=CH2", name = "2,3,3,3-TETRAFLUOROPROPENE", mw = 114.0416, Tc = 367.8500, Pc = 3382200.00, Zc = 0.265190, acf = 0.↩ 276000, Tb = 243.7000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.67996000e+01, 2.41112000e+03, -6.30281000e+00/), Tantmin = 243.7000, Tantmax = 366.0700, Zra = 0.264500, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp64** = cpdata(cid = "R1234yf", ref = "Default", bib_ref = "", cptype = 6, cp = (/2.↩ 13968000e-01,2.10720000e-03,1.89670000e-06,-6.68177000e-09,4.25854000e-12, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 170.0000, Tcpmax = 1000.0000)
- type(alphadatadb), parameter **twu105** = alphadatadb(eosid="PR", cid="R1234yf", ref="tcPR", coeff=(/1.↩ 71200000e-01, 8.37400000e-01, 2.58130000e+00/))
- type(cidatadb), parameter **c106** = cidatadb(eosid="PR", cid="R1234yf", ref="tcPR", bib_ref="10.1016/j.↩ fluid.2016.09.003", ciA=4.91000000e-07, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu106** = alphadatadb(eosid="SRK", cid="R1234yf", ref="tcRK", coeff=(/2.↩ 42900000e-01, 8.45500000e-01, 2.62360000e+00/))

- type(cidatadb), parameter **c107** = cidatadb(eosid="SRK", cid="R1234yf", ref="tcRK", bib_ref="10.1016/j.↩ fluid.2016.09.003", ciA=1.42961000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx64** = gendatadb(ident = "R1234ze", formula = "CHF=CHCF3_(t", name = "TRANS-1,3,3,3-TETRAFLUOROPROPENE", mw = 114.0416, Tc = 382.5130, Pc = 3634900.00, Zc = 0.↩ 266412, acf = 0.313000, Tb = 254.1770, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.↩ 0000, DfG = 0.0000, psatcode = 1, ant = (/1.69543000e+01, 2.50917000e+03, -1.08418000e+01/), Tantmin = 253.8000, Tantmax = 383.0200, Zra = 0.266400, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp65** = cpdata(cid = "R1234ze", ref = "Default", bib_ref = "", cptype = 6, cp = (/-1.↩ 75392000e-01,7.96320000e-03,-2.60047000e-05,4.67071000e-08,-3.17226000e-11, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 170.0000, Tcpmax = 500.↩ 0000)
- type(alphadatadb), parameter **twu107** = alphadatadb(eosid="PR", cid="R1234ze", ref="tcPR", coeff=(/1.↩ 47200000e-01, 8.30400000e-01, 2.88900000e+00/))
- type(cidatadb), parameter **c108** = cidatadb(eosid="PR", cid="R1234ze", ref="tcPR", bib_ref="10.1016/j.↩ fluid.2016.09.003", ciA=-1.47890000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu108** = alphadatadb(eosid="SRK", cid="R1234ze", ref="tcRK", coeff=(/2.↩ 23900000e-01, 8.33600000e-01, 2.83390000e+00/))
- type(cidatadb), parameter **c109** = cidatadb(eosid="SRK", cid="R1234ze", ref="tcRK", bib_ref="10.1016/j.↩ fluid.2016.09.003", ciA=1.16861000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx65** = gendatadb(ident = "R124", formula = "C2HCLF4", name = "2-CHLORO-1,1,1,2-TETRAFLUOROETHANE", mw = 136.4750, Tc = 395.4000, Pc = 3620000.00, Zc = 0.266000, acf = 0.288000, Tb = 261.1000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.269700, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp66** = cpdata(cid = "R124", ref = "Default", bib_ref = "", cptype = 5, cp = (/4.↩ 56447000e-01,1.78778000e-03,3.17361700e-08,-2.23347000e-11,-3.56841800e+01, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu109** = alphadatadb(eosid="PR", cid="R124", ref="tcPR", coeff=(/2.↩ 00900000e-01, 8.50200000e-01, 2.54620000e+00/))
- type(cidatadb), parameter **c110** = cidatadb(eosid="PR", cid="R124", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-3.07770000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu110** = alphadatadb(eosid="SRK", cid="R124", ref="tcRK", coeff=(/2.↩ 64900000e-01, 8.49600000e-01, 2.55500000e+00/))
- type(cidatadb), parameter **c111** = cidatadb(eosid="SRK", cid="R124", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.06893000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx66** = gendatadb(ident = "R124a", formula = "C2HCLF4", name = "1-CHLORO-1,1,2,2-TETRAFLUOROETHANE", mw = 136.4750, Tc = 399.9000, Pc = 3720000.00, Zc = 0.273000, acf = 0.281000, Tb = 263.0000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = -1.000000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp67** = cpdata(cid = "R124a", ref = "Default", bib_ref = "", cptype = 4, cp = (/1.↩ 27000000e+01,3.79500000e-01,-3.46000000e-04,1.15400000e-07,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(gendatadb), parameter **cx67** = gendatadb(ident = "R125", formula = "C2HF5", name = "PENTAFLUOROETHANE", mw = 120.0300, Tc = 343.7000, Pc = 3870000.00, Zc = 0.298600, acf = 0.269000, Tb = 224.6500, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.267100, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp68** = cpdata(cid = "R125", ref = "Default", bib_ref = "", cptype = 4, cp = (/5.↩ 37000000e+00,3.84500000e-01,-3.42000000e-04,1.12000000e-07,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu111** = alphadatadb(eosid="PR", cid="R125", ref="tcPR", coeff=(/1.↩ 78000000e-01, 8.41400000e-01, 2.69360000e+00/))

- type(cidatadb), parameter **c112** = cidatadb(eosid="PR", cid="R125", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-1.45020000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu112** = alphadatadb(eosid="SRK", cid="R125", ref="tcRK", coeff=(/2.↩ 64000000e-01, 8.46300000e-01, 2.60960000e+00/))
- type(cidatadb), parameter **c113** = cidatadb(eosid="SRK", cid="R125", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.04040000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx68** = gendatadb(ident = "R13", formula = "CCLF3", name = "CHLOROTRI-FLUOROMETHANE", mw = 104.4590, Tc = 302.0000, Pc = 3921300.00, Zc = 0.282000, acf = 0.180000, Tb = 191.7000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.277100, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp69** = cpdata(cid = "R13", ref = "Default", bib_ref = "", cptype = 1, cp = (/5.↩ 44900000e+00,4.56500000e-02,-3.76500000e-05,1.06500000e-08,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu113** = alphadatadb(eosid="PR", cid="R13", ref="tcPR", coeff=(/1.↩ 40000000e-01, 8.58100000e-01, 2.42490000e+00/))
- type(cidatadb), parameter **c114** = cidatadb(eosid="PR", cid="R13", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-3.88590000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu114** = alphadatadb(eosid="SRK", cid="R13", ref="tcRK", coeff=(/2.↩ 15200000e-01, 8.60900000e-01, 2.38380000e+00/))
- type(cidatadb), parameter **c115** = cidatadb(eosid="SRK", cid="R13", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=6.25440000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx69** = gendatadb(ident = "R134a", formula = "C2H2F4", name = "1,1,1,2-TETRAFLUOROETHANE", mw = 102.0300, Tc = 374.1790, Pc = 4056000.00, Zc = 0.259100, acf = 0.↩ 326680, Tb = 246.7000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.259600, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp70** = cpdata(cid = "R134a", ref = "Default", bib_ref = "", cptype = 5, cp = (/1.↩ 31419000e-01,3.00600000e-03,-2.23892000e-06,5.97826000e-10,4.30007700e+02, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu115** = alphadatadb(eosid="PR", cid="R134a", ref="tcPR", coeff=(/2.↩ 29200000e-01, 8.50000000e-01, 2.54990000e+00/))
- type(cidatadb), parameter **c116** = cidatadb(eosid="PR", cid="R134a", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=2.19800000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu116** = alphadatadb(eosid="SRK", cid="R134a", ref="tcRK", coeff=(/3.↩ 22600000e-01, 8.56200000e-01, 2.45240000e+00/))
- type(cidatadb), parameter **c117** = cidatadb(eosid="SRK", cid="R134a", ref="tcRK", bib_ref="10.1016/j.↩ fluid.2016.09.003", ciA=1.38434000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx70** = gendatadb(ident = "R14", formula = "CF4", name = "CARBON TETRAFLUORIDE", mw = 88.0050, Tc = 227.6000, Pc = 3738900.00, Zc = 0.277000, acf = 0.191000, Tb = 145.2000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.60543000e+01, 1.24455000e+03, -1.30600000e+01/), Tantmin = 93.0000, Tantmax = 148.0000, Zra = 0.281000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp71** = cpdata(cid = "R14", ref = "Default", bib_ref = "", cptype = 1, cp = (/3.↩ 33900000e+00,4.83800000e-02,-3.88300000e-05,1.07800000e-08,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu117** = alphadatadb(eosid="PR", cid="R14", ref="tcPR", coeff=(/1.↩ 65300000e-01, 8.58400000e-01, 2.29530000e+00/))
- type(cidatadb), parameter **c118** = cidatadb(eosid="PR", cid="R14", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-4.52580000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu118** = alphadatadb(eosid="SRK", cid="R14", ref="tcRK", coeff=(/2.↩ 40400000e-01, 8.65900000e-01, 2.31260000e+00/))
- type(cidatadb), parameter **c119** = cidatadb(eosid="SRK", cid="R14", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=3.34870000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)

- type(gendatadb), parameter **cx71** = gendatadb(ident = "R142b", formula = "C2H3ClF2", name = "1-CHLORO-1,1-DIFLUOROETHANE", mw = 100.4960, Tc = 409.6000, Pc = 4218000.00, Zc = 0.286100, acf = 0.236000, Tb = 262.9300, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.266800, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp72** = cpdata(cid = "R142b", ref = "Default", bib_ref = "", cptype = 1, cp = (/4.↩ 01700000e+00,6.58400000e-02,-4.75800000e-05,1.26700000e-08,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu119** = alphadatadb(eosid="PR", cid="R142b", ref="tcPR", coeff=(/1.↩ 78700000e-01, 8.55600000e-01, 2.46250000e+00/))
- type(cidatadb), parameter **c120** = cidatadb(eosid="PR", cid="R142b", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-2.19000000e-08, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu120** = alphadatadb(eosid="SRK", cid="R142b", ref="tcRK", coeff=(/2.↩ 56100000e-01, 8.58500000e-01, 2.41880000e+00/))
- type(cidatadb), parameter **c121** = cidatadb(eosid="SRK", cid="R142b", ref="tcRK", bib_ref="10.1016/j.↩ fluid.2016.09.003", ciA=1.30216000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx72** = gendatadb(ident = "R143a", formula = "C2H3F3", name = "1,1,1-TRIFLUOROETHANE", mw = 84.0410, Tc = 346.3000, Pc = 3760000.00, Zc = 0.253000, acf = 0.251000, Tb = 225.6000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.256700, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp73** = cpdata(cid = "R143a", ref = "Default", bib_ref = "", cptype = 4, cp = (/5.↩ 74400000e+00,3.14100000e-01,-2.59700000e-04,8.41500000e-08,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu121** = alphadatadb(eosid="PR", cid="R143a", ref="tcPR", coeff=(/2.↩ 03700000e-01, 8.56000000e-01, 2.45580000e+00/))
- type(cidatadb), parameter **c122** = cidatadb(eosid="PR", cid="R143a", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=4.78330000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu122** = alphadatadb(eosid="SRK", cid="R143a", ref="tcRK", coeff=(/2.↩ 95700000e-01, 8.62300000e-01, 2.36330000e+00/))
- type(cidatadb), parameter **c123** = cidatadb(eosid="SRK", cid="R143a", ref="tcRK", bib_ref="10.1016/j.↩ fluid.2016.09.003", ciA=1.65099000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx73** = gendatadb(ident = "R152a", formula = "C2H4F2", name = "1,1-DIFLUOROETHANE", mw = 66.0510, Tc = 386.6000, Pc = 4498800.00, Zc = 0.253000, acf = 0.266000, Tb = 248.4000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.61871000e+01, 2.09535000e+03, -2.91600000e+01/), Tantmin = 238.0000, Tantmax = 273.0000, Zra = 0.253800, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp74** = cpdata(cid = "R152a", ref = "Default", bib_ref = "", cptype = 1, cp = (/2.↩ 07200000e+00,5.72200000e-02,-3.48000000e-05,8.10700000e-09,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu123** = alphadatadb(eosid="PR", cid="R152a", ref="tcPR", coeff=(/2.↩ 77700000e-01, 8.73100000e-01, 2.21630000e+00/))
- type(cidatadb), parameter **c124** = cidatadb(eosid="PR", cid="R152a", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=5.59180000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu124** = alphadatadb(eosid="SRK", cid="R152a", ref="tcRK", coeff=(/3.↩ 87500000e-01, 8.83100000e-01, 2.09120000e+00/))
- type(cidatadb), parameter **c125** = cidatadb(eosid="SRK", cid="R152a", ref="tcRK", bib_ref="10.1016/j.↩ fluid.2016.09.003", ciA=1.64932000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx74** = gendatadb(ident = "R21", formula = "CHCL2F", name = "DICHLOROFLUOROMETHANE", mw = 102.9230, Tc = 451.6000, Pc = 5167600.00, Zc = 0.272000, acf = 0.202000, Tb = 282.0000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.270500, mu_dipole = 0.000000, q_quadrupole = 0.000000)

- type(cpdata), parameter **cp75** = cpdata(cid = "R21", ref = "Default", bib_ref = "", cptype = 1, cp = (/5.↩
65200000e+00,3.77700000e-02,-2.86600000e-05,7.79500000e-09,0.00000000e+00, 0.00000000e+00,0.↩
00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000
)
- type(alphadatadb), parameter **twu125** = alphadatadb(eosid="PR", cid="R21", ref="tcPR", coeff=(/1.↩
44100000e-01, 8.51500000e-01, 2.52500000e+00/))
- type(cidatadb), parameter **c126** = cidatadb(eosid="PR", cid="R21", ref="tcPR", bib_ref="10.1016/j.fluid.↩
2016.09.003", ciA=-1.78500000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu126** = alphadatadb(eosid="SRK", cid="R21", ref="tcRK", coeff=(/2.↩
04900000e-01, 8.50900000e-01, 2.53470000e+00/))
- type(cidatadb), parameter **c127** = cidatadb(eosid="SRK", cid="R21", ref="tcRK", bib_ref="10.1016/j.fluid.↩
2016.09.003", ciA=9.50530000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx75** = gendatadb(ident = "R218", formula = "C3F8", name = "OCTAFLUOROPROPANE", mw = 188.0170, Tc = 345.1000, Pc = 2680000.00, Zc = 0.280000, acf = 0.325000, Tb = 236.5000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.277800, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp76** = cpdata(cid = "R218", ref = "Default", bib_ref = "", cptype = 4, cp = (/1.↩
29400000e+01,6.22000000e-01,-6.40800000e-04,2.39800000e-07,0.00000000e+00, 0.00000000e+00,0.↩
00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000
)
- type(alphadatadb), parameter **twu127** = alphadatadb(eosid="PR", cid="R218", ref="tcPR", coeff=(/1.↩
03840000e+00, 1.00000000e+00, 8.04600000e-01/))
- type(cidatadb), parameter **c128** = cidatadb(eosid="PR", cid="R218", ref="tcPR", bib_ref="10.1016/j.fluid.↩
2016.09.003", ciA=-1.06166000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu128** = alphadatadb(eosid="SRK", cid="R218", ref="tcRK", coeff=(/8.↩
84000000e-01, 9.71000000e-01, 1.11890000e+00/))
- type(cidatadb), parameter **c129** = cidatadb(eosid="SRK", cid="R218", ref="tcRK", bib_ref="10.1016/j.fluid.↩
2016.09.003", ciA=5.58080000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx76** = gendatadb(ident = "R22", formula = "CHCLF2", name = "CHLORODIFLUOROMETHANE", mw = 86.4690, Tc = 369.2000, Pc = 4975100.00, Zc = 0.267000, acf = 0.215000, Tb = 232.4000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.55602000e+01, 1.70480000e+03, -4.13000000e+01/), Tantmin = 225.0000, Tantmax = 240.0000, Zra = 0.266300, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp77** = cpdata(cid = "R22", ref = "Default", bib_ref = "", cptype = 1, cp = (/4.↩
13200000e+00,3.86500000e-02,-2.79400000e-05,7.30500000e-09,0.00000000e+00, 0.00000000e+00,0.↩
00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000
)
- type(alphadatadb), parameter **twu129** = alphadatadb(eosid="PR", cid="R22", ref="tcPR", coeff=(/4.↩
51300000e-01, 8.26700000e-01, 1.24430000e+00/))
- type(cidatadb), parameter **c130** = cidatadb(eosid="PR", cid="R22", ref="tcPR", bib_ref="10.1016/j.fluid.↩
2016.09.003", ciA=-1.18800000e-07, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu130** = alphadatadb(eosid="SRK", cid="R22", ref="tcRK", coeff=(/4.↩
03800000e-01, 8.40800000e-01, 1.63470000e+00/))
- type(cidatadb), parameter **c131** = cidatadb(eosid="SRK", cid="R22", ref="tcRK", bib_ref="10.1016/j.fluid.↩
2016.09.003", ciA=9.50070000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx77** = gendatadb(ident = "R23", formula = "CHF3", name = "TRIFLUOROMETHANE", mw = 70.0130, Tc = 299.3000, Pc = 4860000.00, Zc = 0.259000, acf = 0.260000, Tb = 191.0000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.257600, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp78** = cpdata(cid = "R23", ref = "Default", bib_ref = "", cptype = 4, cp = (/8.↩
15600000e+00,1.81300000e-01,-1.37900000e-04,3.93800000e-08,0.00000000e+00, 0.00000000e+00,0.↩
00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000
)
- type(alphadatadb), parameter **twu131** = alphadatadb(eosid="PR", cid="R23", ref="tcPR", coeff=(/3.↩
86400000e-01, 8.42000000e-01, 1.57440000e+00/))

- type(cidatadb), parameter **c132** = cidatadb(eosid="PR", cid="R23", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=2.94380000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu132** = alphadatadb(eosid="SRK", cid="R23", ref="tcRK", coeff=(/3.↩ 60900000e-01, 8.53700000e-01, 2.00220000e+00/))
- type(cidatadb), parameter **c133** = cidatadb(eosid="SRK", cid="R23", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.08742000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx78** = gendatadb(ident = "R32", formula = "CH2F2", name = "DIFLUOROMETHANE", mw = 52.0230, Tc = 351.6000, Pc = 5830000.00, Zc = 0.241000, acf = 0.271000, Tb = 221.5000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.244400, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp79** = cpdata(cid = "R32", ref = "Default", bib_ref = "", cptype = 4, cp = (/1.↩ 17900000e+01,1.18100000e-01,-4.84300000e-05,2.12500000e-09,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu133** = alphadatadb(eosid="PR", cid="R32", ref="tcPR", coeff=(/2.↩ 48300000e-01, 8.64400000e-01, 2.33320000e+00/))
- type(cidatadb), parameter **c134** = cidatadb(eosid="PR", cid="R32", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=7.19530000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu134** = alphadatadb(eosid="SRK", cid="R32", ref="tcRK", coeff=(/3.↩ 48300000e-01, 8.72400000e-01, 2.22590000e+00/))
- type(cidatadb), parameter **c135** = cidatadb(eosid="SRK", cid="R32", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.49356000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx79** = gendatadb(ident = "R41", formula = "CH3F", name = "METHYL FLUORIDE", mw = 34.0330, Tc = 315.0000, Pc = 5600000.00, Zc = 0.240000, acf = 0.187000, Tb = 194.7000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.↩ 00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.248100, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp80** = cpdata(cid = "R41", ref = "Default", bib_ref = "", cptype = 4, cp = (/1.38200000e+01,8.61600000e-02,-2.07100000e-05,-1.98500000e-09,0.00000000e+00, 0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu135** = alphadatadb(eosid="PR", cid="R41", ref="tcPR", coeff=(/2.↩ 56600000e-01, 8.73000000e-01, 1.96820000e+00/))
- type(cidatadb), parameter **c136** = cidatadb(eosid="PR", cid="R41", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=6.15490000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu136** = alphadatadb(eosid="SRK", cid="R41", ref="tcRK", coeff=(/2.↩ 95400000e-01, 8.77000000e-01, 2.16570000e+00/))
- type(cidatadb), parameter **c137** = cidatadb(eosid="SRK", cid="R41", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.31619000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx80** = gendatadb(ident = "F6S", formula = "F6S", name = "SULFUR HEXAFLUORIDE", mw = 146.0540, Tc = 318.7000, Pc = 3760000.00, Zc = 0.282000, acf = 0.286000, Tb = 209.6000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.↩ 278800, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp81** = cpdata(cid = "F6S", ref = "Default", bib_ref = "", cptype = 4, cp = (/-6.↩ 59900000e-01,4.63900000e-01,-5.08900000e-04,1.95300000e-07,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu137** = alphadatadb(eosid="PR", cid="F6S", ref="tcPR", coeff=(/4.↩ 93500000e-01, 4.81600000e-01, 8.17500000e-01/))
- type(cidatadb), parameter **c138** = cidatadb(eosid="PR", cid="F6S", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-5.49410000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu138** = alphadatadb(eosid="SRK", cid="F6S", ref="tcRK", coeff=(/1.↩ 05890000e+00, 8.35400000e-01, 7.30700000e-01/))
- type(cidatadb), parameter **c139** = cidatadb(eosid="SRK", cid="F6S", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=5.39720000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)

- type(gendatadb), parameter **cx81** = gendatadb(ident = "SO2", formula = "SO2", name = "SULFUR DIOXIDE", mw = 64.0650, Tc = 430.8000, Pc = 7885000.00, Zc = 0.269000, acf = 0.251000, Tb = 263.1300, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.↩ 67681000e+01, 2.30240000e+03, -3.59600000e+01/), Tantmin = 199.7100, Tantmax = 279.4700, Zra = 0.↩ 266100, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp82** = cpdata(cid = "SO2", ref = "Default", bib_ref = "", cptype = 6, cp = (/5.↩ 73200000e-01,-2.89930000e-04,3.04210000e-06,-4.24520000e-09,1.80790000e-12, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 50.0000, Tcpmax = 1000.↩ 0000)
- type(alphadatadb), parameter **twu139** = alphadatadb(eosid="PR", cid="SO2", ref="tcPR", coeff=(/4.↩ 18400000e-01, 8.23800000e-01, 1.40680000e+00/))
- type(cidatadb), parameter **c140** = cidatadb(eosid="PR", cid="SO2", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-4.93000000e-08, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu140** = alphadatadb(eosid="SRK", cid="SO2", ref="tcRK", coeff=(/4.↩ 01400000e-01, 8.35800000e-01, 1.73550000e+00/))
- type(cidatadb), parameter **c141** = cidatadb(eosid="SRK", cid="SO2", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=6.99930000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx82** = gendatadb(ident = "F4N2", formula = "F4N2", name = "TETRAFLUOROHYDRAZINE", mw = 104.0160, Tc = 309.3000, Pc = 3750000.00, Zc = 0.000000, acf = 0.206000, Tb = 199.0000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = -1.000000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp83** = cpdata(cid = "F4N2", ref = "Default", bib_ref = "", cptype = 4, cp = (/3.↩ 55300000e+00,3.50900000e-01,-3.63700000e-04,1.33800000e-07,0.00000000e+00, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(alphadatadb), parameter **twu141** = alphadatadb(eosid="PR", cid="F4N2", ref="tcPR", coeff=(/4.↩ 61700000e-01, 9.45500000e-01, 1.48230000e+00/))
- type(cidatadb), parameter **c142** = cidatadb(eosid="PR", cid="F4N2", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=0.00000000e+00, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu142** = alphadatadb(eosid="SRK", cid="F4N2", ref="tcRK", coeff=(/5.↩ 05000000e-01, 9.34500000e-01, 1.57380000e+00/))
- type(cidatadb), parameter **c143** = cidatadb(eosid="SRK", cid="F4N2", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=0.00000000e+00, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx83** = gendatadb(ident = "TOLU", formula = "C7H8", name = "TOLUENE", mw = 92.1410, Tc = 591.7900, Pc = 4108600.00, Zc = 0.264000, acf = 0.264100, Tb = 383.7800, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.264300, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp84** = cpdata(cid = "TOLU", ref = "Default", bib_ref = "", cptype = 7, cp = (/5.81400000e+04,2.86300000e+05,1.44060000e+03,1.89800000e+05,-6.50430000e+02, 0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 200.↩ 0000, Tcpmax = 1500.0000)
- type(alphadatadb), parameter **mc42** = alphadatadb(eosid="PR", cid="TOLU", ref="Chapoy2005", coeff=(/7.↩ 62000000e-01, -4.20000000e-02, 2.71000000e-01/))
- type(alphadatadb), parameter **twu143** = alphadatadb(eosid="PR", cid="TOLU", ref="tcPR", coeff=(/3.↩ 09400000e-01, 8.30500000e-01, 1.78080000e+00/))
- type(cidatadb), parameter **c144** = cidatadb(eosid="PR", cid="TOLU", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.23690000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **mc43** = alphadatadb(eosid="SRK", cid="TOLU", ref="Chapoy2005", coeff=(/9.23000000e-01, -3.01000000e-01, 4.94000000e-01/))
- type(alphadatadb), parameter **twu144** = alphadatadb(eosid="SRK", cid="TOLU", ref="tcRK", coeff=(/3.↩ 25200000e-01, 8.43800000e-01, 2.10000000e+00/))
- type(cidatadb), parameter **c145** = cidatadb(eosid="SRK", cid="TOLU", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.96684000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)

- type(gendatadb), parameter **cx84** = gendatadb(ident = "F3NO", formula = "F3N0", name = "TRIFLUOROAMINEOXIDE", mw = 87.0010, Tc = 303.0000, Pc = 6430000.00, Zc = 0.375000, acf = 0.212000, Tb = 186.0000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = -1.000000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp85** = cpdata(cid = "F3NO", ref = "Default", bib_ref = "", cptype = 4, cp = (/1.51300000e+01,2.44600000e-01,-2.52800000e-04,9.37500000e-08,0.00000000e+00, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(gendatadb), parameter **cx85** = gendatadb(ident = "H2O", formula = "H2O", name = "WATER", mw = 18.0150, Tc = 647.3000, Pc = 22048300.00, Zc = 0.229000, acf = 0.344000, Tb = 373.2000, Ttr = 0.0000, Ptr = 0.0000, sref = 188.8400, href = -241826.4000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.83036000e+01, 3.81644000e+03, -4.61300000e+01/), Tantmin = 284.0000, Tantmax = 441.0000, Zra = 0.233800, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp86** = cpdata(cid = "H2O", ref = "Default", bib_ref = "", cptype = 2, cp = (/-5.72991500e+00,1.91500700e+00,-3.95741000e-04,8.76232000e-07,-4.95086000e-10, 1.03861300e-13,7.02815000e-01,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -175.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu145** = alphadatadb(eosid="PR", cid="H2O", ref="tcPR", coeff=(/3.86500000e-01, 8.72000000e-01, 1.96930000e+00/))
- type(alphadatadb), parameter **mc44** = alphadatadb(eosid="PR", cid="H2O", ref="Chapoy2005", coeff=(/9.19000000e-01, -3.32000000e-01, 3.17000000e-01/))
- type(cidatadb), parameter **c146** = cidatadb(eosid="PR", cid="H2O", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=5.30410000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cidatadb), parameter **c147** = cidatadb(eosid="PR", cid="H2O", ref="tcPR-ENGINEERING", bib_ref="", ciA=-8.12803800e-08, ciB=1.04455800e-08, ciC=0.00000000e+00, c_type=2)
- type(alphadatadb), parameter **twu146** = alphadatadb(eosid="SRK", cid="H2O", ref="tcRK", coeff=(/4.16300000e-01, 8.75600000e-01, 2.18420000e+00/))
- type(alphadatadb), parameter **mc45** = alphadatadb(eosid="SRK", cid="H2O", ref="Chapoy2005", coeff=(/1.09500000e+00, -6.78000000e-01, 7.00000000e-01/))
- type(cidatadb), parameter **c148** = cidatadb(eosid="SRK", cid="H2O", ref="tcRK", bib_ref="10.1016/j.fluid.2016.09.003", ciA=8.99950000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cpadata), parameter **cpa13** = CPAdata(eosid="CPA-SRK", compName="H2O", ref="Default/Queimada2005", bib_reference="10.1016/j.fluid.2004.08.011", a0=1.22770000e+05, b=1.45150000e-02, eps=1.66550000e+04, beta=6.92000000e-02, alphacorridx = cbAlphaClassicIdx, alphaParams = (/6.73590000e-01,0.00000000e+00,0.00000000e+00/), assoc_scheme = assoc_scheme_4C)
- type(cpadata), parameter **cpa14** = CPAdata(eosid="CPA-SRK", compName="H2O", ref="SINTEF", bib_reference="", a0=4.67542000e+05, b=1.57983000e-02, eps=4.44953000e+03, beta=6.21918000e-03, alphacorridx = cbAlphaClassicIdx, alphaParams = (/7.76671000e-01,0.00000000e+00,0.00000000e+00/), assoc_scheme = assoc_scheme_2B)
- type(gendatadb), parameter **cx86** = gendatadb(ident = "XE", formula = "XE", name = "XENON", mw = 131.3000, Tc = 289.7000, Pc = 5840000.00, Zc = 0.287000, acf = 0.008000, Tb = 165.0000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 2, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.282900, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp87** = cpdata(cid = "XE", ref = "Default", bib_ref = "", cptype = 4, cp = (/2.07860000e+01,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 0.0000, Tcpmax = 0.0000)
- type(gendatadb), parameter **cx87** = gendatadb(ident = "NC4", formula = "C4H10", name = "N-BUTANE", mw = 58.1240, Tc = 425.2000, Pc = 3799700.00, Zc = 0.274000, acf = 0.193000, Tb = 272.7000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.56782000e+01, 2.15490000e+03, -3.44200000e+01/), Tantmin = 195.0000, Tantmax = 290.0000, Zra = 0.273000, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp88** = cpdata(cid = "NC4", ref = "Default", bib_ref = "", cptype = 2, cp = (/1.72831340e+01,4.12696000e-01,2.02860100e-03,7.02953000e-07,-1.02587100e-09, 2.88339400e-13,2.71486100e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -75.0000, Tcpmax = 1200.0000)

- type(alphadatadb), parameter **mc46** = alphadatadb(eosid="PR", cid="NC4", ref="Default", coeff=(/8.↩ 78700000e-01, -9.39900000e-01, 2.26660000e+00/))
- type(alphadatadb), parameter **mc47** = alphadatadb(eosid="PR", cid="NC4", ref="Chapoy2005", coeff=(/6.↩ 77000000e-01, -8.10000000e-02, 2.99000000e-01/))
- type(alphadatadb), parameter **twu147** = alphadatadb(eosid="PR", cid="NC4", ref="tcPR", coeff=(/1.↩ 86700000e-01, 8.64500000e-01, 2.33270000e+00/))
- type(cidatadb), parameter **c149** = cidatadb(eosid="PR", cid="NC4", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=-3.58180000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **mc48** = alphadatadb(eosid="SRK", cid="NC4", ref="Default", coeff=(/8.↩ 78700000e-01, -9.39900000e-01, 2.26660000e+00/))
- type(alphadatadb), parameter **mc49** = alphadatadb(eosid="SRK", cid="NC4", ref="Chapoy2005", coeff=(/8.23000000e-01, -2.67000000e-01, 4.02000000e-01/))
- type(alphadatadb), parameter **twu148** = alphadatadb(eosid="SRK", cid="NC4", ref="tcRK", coeff=(/2.↩ 62100000e-01, 8.66900000e-01, 2.29960000e+00/))
- type(cidatadb), parameter **c150** = cidatadb(eosid="SRK", cid="NC4", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.09178000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cpadata), parameter **cpa15** = CPAdata(eosid="CPA-SRK", compName="NC4", ref="Default/Kontogeorgis-Folas2010", bib_reference="10.1002/9780470747537", a0=1.31427400e+06, b=7.20810000e-02, eps=0.↩ 00000000e+00, beta=0.00000000e+00, alphacorridx = cbAlphaClassicIdx, alphaParams = (/7.07710000e-01,0.00000000e+00,0.00000000e+00/), assoc_scheme = no_assoc)
- type(gendatadb), parameter **cx88** = gendatadb(ident = "NC10", formula = "C10H22", name = "N-DECANE", mw = 142.2860, Tc = 617.6000, Pc = 2107600.00, Zc = 0.247000, acf = 0.490000, Tb = 447.3000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.60114000e+01, 3.45680000e+03, -7.86700000e+01/), Tantmin = 330.0000, Tantmax = 476.0000, Zra = 0.250700, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp89** = cpdata(cid = "NC10", ref = "Default", bib_ref = "", cptype = 2, cp = (/-6.96202000e+00,8.51375000e-01,-2.63041000e-04,5.52181600e-06,-5.63173300e-09, 1.88854430e-12,-4.12446000e-01,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -75.0000, Tcpmax = 700.0000)
- type(alphadatadb), parameter **twu149** = alphadatadb(eosid="PR", cid="NC10", ref="tcPR", coeff=(/3.↩ 67700000e-01, 8.11900000e-01, 2.21880000e+00/))
- type(cidatadb), parameter **c151** = cidatadb(eosid="PR", cid="NC10", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.28105000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu150** = alphadatadb(eosid="SRK", cid="NC10", ref="tcRK", coeff=(/3.↩ 55300000e-01, 8.31000000e-01, 2.72810000e+00/))
- type(cidatadb), parameter **c152** = cidatadb(eosid="SRK", cid="NC10", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=4.85857000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cpadata), parameter **cpa16** = CPAdata(eosid="CPA-SRK", compName="NC10", ref="Default/Kontogeorgis-Folas2010", bib_reference="10.1002/9780470747537", a0=4.73890000e+06, b=1.78650000e-01, eps=0.↩ 00000000e+00, beta=0.00000000e+00, alphacorridx = cbAlphaClassicIdx, alphaParams = (/1.↩ 13243000e+00,0.00000000e+00,0.00000000e+00/), assoc_scheme = no_assoc)
- type(gendatadb), parameter **cx89** = gendatadb(ident = "NC22", formula = "C22H46", name = "N-DOCOSANE", mw = 310.6100, Tc = 787.0000, Pc = 1060000.00, Zc = 0.240000, acf = 0.972200, Tb = 641.7500, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.229950, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp90** = cpdata(cid = "NC22", ref = "Default", bib_ref = "", cptype = 7, cp = (/3.92560000e+05,1.18200000e+06,1.72340000e+03,8.15780000e+05,7.85130000e+02, 0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 300.↩ 0000, Tcpmax = 1500.0000)
- type(alphadatadb), parameter **twu151** = alphadatadb(eosid="PR", cid="NC22", ref="tcPR", coeff=(/4.↩ 78800000e-01, 7.99000000e-01, 2.84990000e+00/))
- type(cidatadb), parameter **c153** = cidatadb(eosid="PR", cid="NC22", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=8.51555000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu152** = alphadatadb(eosid="SRK", cid="NC22", ref="tcRK", coeff=(/5.↩ 01600000e-01, 8.08200000e-01, 3.12430000e+00/))

- type(cidatadb), parameter **c154** = cidatadb(eosid="SRK", cid="NC22", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.71456100e-04, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx90** = gendatadb(ident = "NC12", formula = "C12H26", name = "N-DODECANE", mw = 170.3400, Tc = 658.1000, Pc = 1817000.00, Zc = 0.249700, acf = 0.574000, Tb = 489.3000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.246680, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp91** = cpdata(cid = "NC12", ref = "Default", bib_ref = "", cptype = 8, cp = (/1.↩ 72290000e+01,-7.24200000e-03,3.19220000e-04,-4.23220000e-07,1.70220000e-10, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -20.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu153** = alphadatadb(eosid="PR", cid="NC12", ref="tcPR", coeff=(/3.↩ 95600000e-01, 8.11200000e-01, 2.35490000e+00/))
- type(cidatadb), parameter **c155** = cidatadb(eosid="PR", cid="NC12", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=2.20285000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu154** = alphadatadb(eosid="SRK", cid="NC12", ref="tcRK", coeff=(/3.↩ 88700000e-01, 8.27500000e-01, 2.82130000e+00/))
- type(cidatadb), parameter **c156** = cidatadb(eosid="SRK", cid="NC12", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=6.56626000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx91** = gendatadb(ident = "NC20", formula = "C20H42", name = "N-EICOSANE", mw = 282.5500, Tc = 768.0000, Pc = 1070000.00, Zc = 0.243000, acf = 0.865000, Tb = 616.8400, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.↩ 00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.232780, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp92** = cpdata(cid = "NC20", ref = "Default", bib_ref = "", cptype = 7, cp = (/3.24810000e+05,1.10900000e+06,1.63600000e+03,7.45000000e+05,7.26270000e+02, 0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 200.↩ 0000, Tcpmax = 1500.0000)
- type(alphadatadb), parameter **twu155** = alphadatadb(eosid="PR", cid="NC20", ref="tcPR", coeff=(/4.↩ 77100000e-01, 8.16000000e-01, 2.92090000e+00/))
- type(cidatadb), parameter **c157** = cidatadb(eosid="PR", cid="NC20", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=6.37019000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu156** = alphadatadb(eosid="SRK", cid="NC20", ref="tcRK", coeff=(/5.↩ 21900000e-01, 8.21000000e-01, 3.08880000e+00/))
- type(cidatadb), parameter **c158** = cidatadb(eosid="SRK", cid="NC20", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.40431300e-04, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx92** = gendatadb(ident = "NC21", formula = "C21H44", name = "N-HENEICOSANE", mw = 296.5800, Tc = 778.0000, Pc = 1110000.00, Zc = 0.242000, acf = 0.942000, Tb = 629.6500, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.231530, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp93** = cpdata(cid = "NC21", ref = "Default", bib_ref = "", cptype = 7, cp = (/3.82820000e+05,7.71070000e+05,8.01080000e+02,4.99080000e+05,2.36160000e+03, 0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 300.↩ 0000, Tcpmax = 1500.0000)
- type(alphadatadb), parameter **twu157** = alphadatadb(eosid="PR", cid="NC21", ref="tcPR", coeff=(/4.↩ 54600000e-01, 8.18600000e-01, 3.14140000e+00/))
- type(cidatadb), parameter **c159** = cidatadb(eosid="PR", cid="NC21", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=7.24116000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu158** = alphadatadb(eosid="SRK", cid="NC21", ref="tcRK", coeff=(/5.↩ 30200000e-01, 8.18900000e-01, 3.13490000e+00/))
- type(cidatadb), parameter **c160** = cidatadb(eosid="SRK", cid="NC21", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.53342200e-04, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx93** = gendatadb(ident = "NC17", formula = "C17H36", name = "N-HEPTADECANE", mw = 240.4700, Tc = 736.0000, Pc = 1340000.00, Zc = 0.242000, acf = 0.753000, Tb = 574.5600, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.236420, mu_dipole = 0.000000, q_quadrupole = 0.000000)

- type(cpdata), parameter **cp94** = cpdata(cid = "NC17", ref = "Default", bib_ref = "", cptype = 8, cp = (/2.↩
38130000e+01,-9.21000000e-03,4.53330000e-04,-6.06010000e-07,2.44550000e-10, 0.00000000e+00,0.↩
00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -20.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu159** = alphadatadb(eosid="PR", cid="NC17", ref="tcPR", coeff=(/5.↩
25700000e-01, 7.96900000e-01, 2.30920000e+00/))
- type(cidatadb), parameter **c161** = cidatadb(eosid="PR", cid="NC17", ref="tcPR", bib_ref="10.1016/j.fluid.↩
2016.09.003", ciA=5.24579000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu160** = alphadatadb(eosid="SRK", cid="NC17", ref="tcRK", coeff=(/5.↩
16100000e-01, 8.09800000e-01, 2.69190000e+00/))
- type(cidatadb), parameter **c162** = cidatadb(eosid="SRK", cid="NC17", ref="tcRK", bib_ref="10.1016/j.fluid.↩
2016.09.003", ciA=1.17184700e-04, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx94** = gendatadb(ident = "NC7", formula = "C7H16", name = "N-HEPTANE", mw = 100.2050, Tc = 540.2000, Pc = 2735800.00, Zc = 0.263000, acf = 0.351000, Tb = 371.6000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.58737000e+01, 2.91132000e+03, -5.65100000e+01/), Tantmin = 270.0000, Tantmax = 400.0000, Zra = 0.260400, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp95** = cpdata(cid = "NC7", ref = "Default", bib_ref = "", cptype = 2, cp = (/-1.53725000e-01,7.54499000e-01,2.61728000e-04,4.36635800e-06,-4.48451000e-09, 1.48420990e-12,3.↩
80048000e-01,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -75.0000, Tcpmax = 700.↩
0000)
- type(alphadatadb), parameter **mc50** = alphadatadb(eosid="PR", cid="NC7", ref="Default", coeff=(/1.↩
22780000e+00, -1.55580000e+00, 3.93610000e+00/))
- type(alphadatadb), parameter **mc51** = alphadatadb(eosid="PR", cid="NC7", ref="Chapoy2005", coeff=(/8.↩
78000000e-01, -3.10000000e-02, 3.02000000e-01/))
- type(alphadatadb), parameter **twu161** = alphadatadb(eosid="PR", cid="NC7", ref="tcPR", coeff=(/3.↩
29700000e-01, 8.22200000e-01, 1.96150000e+00/))
- type(cidatadb), parameter **c163** = cidatadb(eosid="PR", cid="NC7", ref="tcPR", bib_ref="10.1016/j.fluid.↩
2016.09.003", ciA=3.09080000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **mc52** = alphadatadb(eosid="SRK", cid="NC7", ref="Default", coeff=(/1.↩
22780000e+00, -1.55580000e+00, 3.93610000e+00/))
- type(alphadatadb), parameter **mc53** = alphadatadb(eosid="SRK", cid="NC7", ref="Chapoy2005", coeff=(/1.03600000e+00, -2.58000000e-01, 4.88000000e-01/))
- type(alphadatadb), parameter **twu162** = alphadatadb(eosid="SRK", cid="NC7", ref="tcRK", coeff=(/3.↩
26900000e-01, 8.38700000e-01, 2.39600000e+00/))
- type(cidatadb), parameter **c164** = cidatadb(eosid="SRK", cid="NC7", ref="tcRK", bib_ref="10.1016/j.fluid.↩
2016.09.003", ciA=2.78263000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cpadata), parameter **cpa17** = CPAdata(eosid="CPA-SRK", compName="NC7", ref="Default/Kontogeorgis-Folas2010", bib_reference="10.1002/9780470747537", a0=2.91780000e+06, b=1.25350000e-01, eps=0.↩
00000000e+00, beta=0.00000000e+00, alphacorridx = cbAlphaClassicIdx, alphaParams = (/9.13700000e-01,0.00000000e+00,0.00000000e+00/), assoc_scheme = no_assoc)
- type(gendatadb), parameter **cx95** = gendatadb(ident = "NC16", formula = "C16H34", name = "N-HEXADECANE", mw = 226.4460, Tc = 717.0000, Pc = 1418600.00, Zc = 0.230000, acf = 0.742000, Tb = 560.0000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.61841000e+01, 4.21491000e+03, -1.18700000e+02/), Tantmin = 423.0000, Tantmax = 594.0000, Zra = 0.238800, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp96** = cpdata(cid = "NC16", ref = "Default", bib_ref = "", cptype = 2, cp = (/6.09270110e+01,-9.55630000e-02,3.45931300e-03,-1.35680700e-06,2.65935000e-10, -1.46753000e-14,3.09512800e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -20.0000, Tcpmax = 1200.0000)
- type(cpdata), parameter **cp97** = cpdata(cid = "NC16", ref = "", bib_ref = "", cptype = 8, cp = (/3.↩
97470000e+01,-2.06152000e-01,1.14814000e-03,-1.55548000e-06,6.75340000e-10, 0.00000000e+00,0.↩
00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -20.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu163** = alphadatadb(eosid="PR", cid="NC16", ref="tcPR", coeff=(/5.↩
37200000e-01, 7.92900000e-01, 2.14580000e+00/))

- type(cidatadb), parameter **c165** = cidatadb(eosid="PR", cid="NC16", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=5.19037000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu164** = alphadatadb(eosid="SRK", cid="NC16", ref="tcRK", coeff=(/5.↩ 26200000e-01, 8.06800000e-01, 2.50800000e+00/))
- type(cidatadb), parameter **c166** = cidatadb(eosid="SRK", cid="NC16", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.13300800e-04, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx96** = gendatadb(ident = "NC6", formula = "C6H14", name = "N-HEXANE", mw = 86.1780, Tc = 507.4000, Pc = 2968800.00, Zc = 0.260000, acf = 0.296000, Tb = 341.9000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.58366000e+01, 2.69755000e+03, -4.87800000e+01/), Tantmin = 245.0000, Tantmax = 370.0000, Zra = 0.263500, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp98** = cpdata(cid = "NC6", ref = "Default", bib_ref = "", cptype = 2, cp = (/-1.71910710e+01,9.59226000e-01,-6.14725000e-04,6.14210100e-06,-6.16095200e-09, 2.08681900e-12,-2.07040000e-01,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -75.0000, Tcpmax = 700.0000)
- type(alphadatadb), parameter **mc54** = alphadatadb(eosid="PR", cid="NC6", ref="Default", coeff=(/1.↩ 04300000e+00, -1.15530000e+00, 2.92350000e+00/))
- type(alphadatadb), parameter **mc55** = alphadatadb(eosid="PR", cid="NC6", ref="Chapoy2005", coeff=(/8.↩ 70000000e-01, -5.88000000e-01, 1.50400000e+00/))
- type(alphadatadb), parameter **twu165** = alphadatadb(eosid="PR", cid="NC6", ref="tcPR", coeff=(/2.↩ 55700000e-01, 8.37700000e-01, 2.18710000e+00/))
- type(cidatadb), parameter **c167** = cidatadb(eosid="PR", cid="NC6", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=7.92500000e-07, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **mc56** = alphadatadb(eosid="SRK", cid="NC6", ref="Default", coeff=(/1.↩ 04300000e+00, -1.15530000e+00, 2.92350000e+00/))
- type(alphadatadb), parameter **mc57** = alphadatadb(eosid="SRK", cid="NC6", ref="Chapoy2005", coeff=(/1.00500000e+00, -5.91000000e-01, 1.20300000e+00/))
- type(alphadatadb), parameter **twu166** = alphadatadb(eosid="SRK", cid="NC6", ref="tcRK", coeff=(/2.↩ 77300000e-01, 8.50300000e-01, 2.54390000e+00/))
- type(cidatadb), parameter **c168** = cidatadb(eosid="SRK", cid="NC6", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=2.20445000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cpadata), parameter **cpa18** = CPAdata(eosid="CPA-SRK", compName="NC6", ref="Default/Kontogeorgis-Folas2010", bib_reference="10.1002/9780470747537", a0=2.36810000e+06, b=1.07890000e-01, eps=0.↩ 00000000e+00, beta=0.00000000e+00, alphacorridx = cbAlphaClassicIdx, alphaParams = (/8.31300000e-01,0.00000000e+00,0.00000000e+00/), assoc_scheme = no_assoc)
- type(gendatadb), parameter **cx97** = gendatadb(ident = "NC19", formula = "C19H40", name = "N-NONADECANE", mw = 268.5300, Tc = 755.0000, Pc = 1160000.00, Zc = 0.242000, acf = 0.845000, Tb = 602.3400, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.233370, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp99** = cpdata(cid = "NC19", ref = "Default", bib_ref = "", cptype = 8, cp = (/2.↩ 64470000e+01,-9.99800000e-03,5.06970000e-04,-6.79120000e-07,2.74280000e-10, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -20.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu167** = alphadatadb(eosid="PR", cid="NC19", ref="tcPR", coeff=(/5.↩ 94600000e-01, 7.93400000e-01, 2.26540000e+00/))
- type(cidatadb), parameter **c169** = cidatadb(eosid="PR", cid="NC19", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=6.49064000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu168** = alphadatadb(eosid="SRK", cid="NC19", ref="tcRK", coeff=(/6.↩ 08600000e-01, 8.02400000e-01, 2.52710000e+00/))
- type(cidatadb), parameter **c170** = cidatadb(eosid="SRK", cid="NC19", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.37989500e-04, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx98** = gendatadb(ident = "NC9", formula = "C9H20", name = "N-NONANE", mw = 128.2590, Tc = 594.6000, Pc = 2310200.00, Zc = 0.260000, acf = 0.444000, Tb = 424.0000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.59671000e+01, 3.29145000e+03, -7.13300000e+01/), Tantmin = 312.0000, Tantmax = 452.0000, Zra = 0.254300, mu_dipole = 0.000000, q_quadrupole = 0.000000)

- type(cpdata), parameter **cp100** = cpdata(cid = "NC9", ref = "Default", bib_ref = "", cptype = 2, cp = (/4.↩ 00027800e+00,7.07805000e-01,4.38048000e-04,3.96934200e-06,-4.04315800e-09, 1.28760280e-12,2.↩ 57265000e-01,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -75.0000, Tcpmax = 700.↩ 0000)
- type(alphadatadb), parameter **twu169** = alphadatadb(eosid="PR", cid="NC9", ref="tcPR", coeff=(/4.↩ 05400000e-01, 8.09700000e-01, 1.93430000e+00/))
- type(cidatadb), parameter **c171** = cidatadb(eosid="PR", cid="NC9", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=9.31890000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu170** = alphadatadb(eosid="SRK", cid="NC9", ref="tcRK", coeff=(/3.↩ 85800000e-01, 8.29400000e-01, 2.40410000e+00/))
- type(cidatadb), parameter **c172** = cidatadb(eosid="SRK", cid="NC9", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=4.13357000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cpadata), parameter **cpa19** = CPAdata(eosid="CPA-SRK", compName="NC9", ref="Default/Kontogeorgis-Folas2010", bib_reference="10.1002/9780470747537", a0=4.12506100e+06, b=1.60350000e-01, eps=0.↩ 00000000e+00, beta=0.00000000e+00, alphacorridx = cbAlphaClassicIdx, alphaParams = (/1.↩ 04628000e+00,0.00000000e+00,0.00000000e+00/), assoc_scheme = no_assoc)
- type(gendatadb), parameter **cx99** = gendatadb(ident = "NC18", formula = "C18H38", name = "N-OCTADECANE", mw = 254.5000, Tc = 747.0000, Pc = 1290000.00, Zc = 0.247000, acf = 0.800000, Tb = 588.3000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.234730, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp101** = cpdata(cid = "NC18", ref = "Default", bib_ref = "", cptype = 8, cp = (/2.↩ 51300000e+01,-9.60300000e-03,4.80150000e-04,-6.42560000e-07,2.59420000e-10, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -20.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu171** = alphadatadb(eosid="PR", cid="NC18", ref="tcPR", coeff=(/5.↩ 53300000e-01, 7.95500000e-01, 2.30670000e+00/))
- type(cidatadb), parameter **c173** = cidatadb(eosid="PR", cid="NC18", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=5.90709000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu172** = alphadatadb(eosid="SRK", cid="NC18", ref="tcRK", coeff=(/5.↩ 50000000e-01, 8.07600000e-01, 2.65400000e+00/))
- type(cidatadb), parameter **c174** = cidatadb(eosid="SRK", cid="NC18", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.27994400e-04, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx100** = gendatadb(ident = "NC8", formula = "C8H18", name = "N-OCTANE", mw = 114.2320, Tc = 568.8000, Pc = 2482500.00, Zc = 0.259000, acf = 0.394000, Tb = 398.8000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.59426000e+01, 3.12029000e+03, -6.36300000e+01/), Tantmin = 292.0000, Tantmax = 425.0000, Zra = 0.257100, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp102** = cpdata(cid = "NC8", ref = "Default", bib_ref = "", cptype = 2, cp = (/2.↩ 60472500e+00,7.24670000e-01,3.67845000e-04,4.14283300e-06,-4.24019900e-09, 1.37340550e-12,3.↩ 27588000e-01,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -75.0000, Tcpmax = 700.↩ 0000)
- type(alphadatadb), parameter **mc58** = alphadatadb(eosid="PR", cid="NC8", ref="Default", coeff=(/1.↩ 27980000e+00, -1.38220000e+00, 3.39330000e+00/))
- type(alphadatadb), parameter **mc59** = alphadatadb(eosid="PR", cid="NC8", ref="Chapoy2005", coeff=(/9.↩ 58000000e-01, -1.34000000e-01, 4.87000000e-01/))
- type(alphadatadb), parameter **twu173** = alphadatadb(eosid="PR", cid="NC8", ref="tcPR", coeff=(/3.↩ 38500000e-01, 8.18500000e-01, 2.07470000e+00/))
- type(cidatadb), parameter **c175** = cidatadb(eosid="PR", cid="NC8", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=6.41340000e-06, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **mc60** = alphadatadb(eosid="SRK", cid="NC8", ref="Default", coeff=(/1.↩ 27980000e+00, -1.38220000e+00, 3.39330000e+00/))
- type(alphadatadb), parameter **mc61** = alphadatadb(eosid="SRK", cid="NC8", ref="Chapoy2005", coeff=(/1.15000000e+00, -5.87000000e-01, 1.09600000e+00/))
- type(alphadatadb), parameter **twu174** = alphadatadb(eosid="SRK", cid="NC8", ref="tcRK", coeff=(/3.↩ 44900000e-01, 8.34100000e-01, 2.46600000e+00/))

- type(cidatadb), parameter **c176** = cidatadb(eosid="SRK", cid="NC8", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=3.48304000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(cpadata), parameter **cpa20** = CPAdata(eosid="CPA-SRK", compName="NC8", ref="Default/Kontogeorgis-Folas2010", bib_reference="10.1002/9780470747537", a0=3.48750000e+06, b=1.42440000e-01, eps=0.↩ 00000000e+00, beta=0.00000000e+00, alphacorridx = cbAlphaClassicIdx, alphaParams = (/9.94150000e-01,0.00000000e+00,0.00000000e+00/), assoc_scheme = no_assoc)
- type(gendatadb), parameter **cx101** = gendatadb(ident = "NC25", formula = "C25H52", name = "N-PENTACOSANE", mw = 352.6900, Tc = 812.0000, Pc = 950000.00, Zc = 0.240000, acf = 1.105300, Tb = 674.1500, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.228110, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp103** = cpdata(cid = "NC25", ref = "Default", bib_ref = "", cptype = 7, cp = (/4.45150000e+05,1.33890000e+06,1.72150000e+03,9.24780000e+05,7.84280000e+02, 0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 300.↩ 0000, Tcpmax = 1500.0000)
- type(alphadatadb), parameter **twu175** = alphadatadb(eosid="PR", cid="NC25", ref="tcPR", coeff=(/5.↩ 71700000e-01, 7.80300000e-01, 2.46660000e+00/))
- type(cidatadb), parameter **c177** = cidatadb(eosid="PR", cid="NC25", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.05664500e-04, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu176** = alphadatadb(eosid="SRK", cid="NC25", ref="tcRK", coeff=(/6.↩ 03400000e-01, 7.79000000e-01, 2.50350000e+00/))
- type(cidatadb), parameter **c178** = cidatadb(eosid="SRK", cid="NC25", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=2.11001600e-04, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx102** = gendatadb(ident = "NC15", formula = "C15H32", name = "N-PENTADECANE", mw = 212.4200, Tc = 708.0000, Pc = 1480000.00, Zc = 0.243000, acf = 0.685000, Tb = 543.8300, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.238360, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp104** = cpdata(cid = "NC15", ref = "Default", bib_ref = "", cptype = 8, cp = (/2.11800000e+01,-8.42400000e-03,3.99690000e-04,-5.30000029e+01,2.14820000e-10, 0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -20.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu177** = alphadatadb(eosid="PR", cid="NC15", ref="tcPR", coeff=(/4.↩ 77000000e-01, 7.97000000e-01, 2.26360000e+00/))
- type(cidatadb), parameter **c179** = cidatadb(eosid="PR", cid="NC15", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=4.53108000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu178** = alphadatadb(eosid="SRK", cid="NC15", ref="tcRK", coeff=(/4.↩ 93500000e-01, 8.08700000e-01, 2.55440000e+00/))
- type(cidatadb), parameter **c180** = cidatadb(eosid="SRK", cid="NC15", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.02313400e-04, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx103** = gendatadb(ident = "NC5", formula = "C5H12", name = "N-PENTAN", mw = 72.1510, Tc = 469.6000, Pc = 3374100.00, Zc = 0.262000, acf = 0.251000, Tb = 309.2000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.58333000e+01, 2.47707000e+03, -3.99400000e+01/), Tantmin = 220.0000, Tantmax = 330.0000, Zra = 0.268400, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp105** = cpdata(cid = "NC5", ref = "Default", bib_ref = "", cptype = 2, cp = (/6.32016770e+01,-1.17010000e-02,3.31649800e-03,-1.17051000e-06,1.99648000e-10, -8.66520000e-15,4.07527500e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -20.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **mc62** = alphadatadb(eosid="PR", cid="NC5", ref="Default", coeff=(/9.↩ 82000000e-01, -1.16950000e+00, 2.75230000e+00/))
- type(alphadatadb), parameter **mc63** = alphadatadb(eosid="PR", cid="NC5", ref="Chapoy2005", coeff=(/7.↩ 63000000e-01, -2.24000000e-01, 6.69000000e-01/))
- type(alphadatadb), parameter **mc64** = alphadatadb(eosid="SRK", cid="NC5", ref="Default", coeff=(/9.↩ 82000000e-01, -1.16950000e+00, 2.75230000e+00/))
- type(alphadatadb), parameter **mc65** = alphadatadb(eosid="SRK", cid="NC5", ref="Chapoy2005", coeff=(/9.01000000e-01, -3.05000000e-01, 5.42000000e-01/))

- type(cpadata), parameter **cpa21** = CPAdata(eosid="CPA-SRK", compName="NC5", ref="Default/Kontogeorgis-Folas2010", bib_reference="10.1002/9780470747537", a0=1.81980000e+06, b=9.10080000e-02, eps=0.00000000e+00, beta=0.00000000e+00, alphacorridx = cbAlphaClassicIdx, alphaParams = (/7.98580000e-01,0.00000000e+00,0.00000000e+00/), assoc_scheme = no_assoc)
- type(gendatadb), parameter **cx104** = gendatadb(ident = "NC14", formula = "C14H30", name = "N-TETRADECANE", mw = 198.3900, Tc = 693.0000, Pc = 1570000.00, Zc = 0.244000, acf = 0.644000, Tb = 526.7600, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.240060, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp106** = cpdata(cid = "NC14", ref = "Default", bib_ref = "", cptype = 8, cp = (/1.83750000e+01,6.58500000e-03,3.23070000e-04,-4.26630000e-07,1.65900000e-10, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -20.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu179** = alphadatadb(eosid="PR", cid="NC14", ref="tcPR", coeff=(/4.90200000e-01, 7.97400000e-01, 2.13530000e+00/))
- type(cidatadb), parameter **c181** = cidatadb(eosid="PR", cid="NC14", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=3.96525000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu180** = alphadatadb(eosid="SRK", cid="NC14", ref="tcRK", coeff=(/4.84100000e-01, 8.11800000e-01, 2.49950000e+00/))
- type(cidatadb), parameter **c182** = cidatadb(eosid="SRK", cid="NC14", ref="tcRK", bib_ref="10.1016/j.fluid.2016.09.003", ciA=9.26542000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx105** = gendatadb(ident = "NC24", formula = "C24H50", name = "N-TETRACOSANE", mw = 338.6600, Tc = 804.0000, Pc = 980000.00, Zc = 0.239000, acf = 1.071000, Tb = 664.4500, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.228390, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp107** = cpdata(cid = "NC24", ref = "Default", bib_ref = "", cptype = 7, cp = (/4.27180000e+05,1.28910000e+06,8.15290000e+02,-5.04180000e+05,9.44980000e+02, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 300.0000, Tcpmax = 1500.0000)
- type(alphadatadb), parameter **twu181** = alphadatadb(eosid="PR", cid="NC24", ref="tcPR", coeff=(/4.65600000e-01, 8.09700000e-01, 3.35130000e+00/))
- type(cidatadb), parameter **c183** = cidatadb(eosid="PR", cid="NC24", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=8.96547000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu182** = alphadatadb(eosid="SRK", cid="NC24", ref="tcRK", coeff=(/5.37300000e-01, 8.09500000e-01, 3.35700000e+00/))
- type(cidatadb), parameter **c184** = cidatadb(eosid="SRK", cid="NC24", ref="tcRK", bib_ref="10.1016/j.fluid.2016.09.003", ciA=1.83043000e-04, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx106** = gendatadb(ident = "NC23", formula = "C23H48", name = "N-TRICOSANE", mw = 324.6300, Tc = 796.0000, Pc = 1020000.00, Zc = 0.240000, acf = 1.026200, Tb = 653.3500, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.229280, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp108** = cpdata(cid = "NC23", ref = "Default", bib_ref = "", cptype = 7, cp = (/4.10150000e+05,1.23420000e+06,1.72310000e+03,8.52280000e+05,7.84970000e+02, 0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = 300.0000, Tcpmax = 1500.0000)
- type(alphadatadb), parameter **twu183** = alphadatadb(eosid="PR", cid="NC23", ref="tcPR", coeff=(/4.69700000e-01, 8.14200000e-01, 3.24260000e+00/))
- type(cidatadb), parameter **c185** = cidatadb(eosid="PR", cid="NC23", ref="tcPR", bib_ref="10.1016/j.fluid.2016.09.003", ciA=8.45878000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu184** = alphadatadb(eosid="SRK", cid="NC23", ref="tcRK", coeff=(/5.44000000e-01, 8.14300000e-01, 3.24000000e+00/))
- type(cidatadb), parameter **c186** = cidatadb(eosid="SRK", cid="NC23", ref="tcRK", bib_ref="10.1016/j.fluid.2016.09.003", ciA=1.73915900e-04, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)

- type(gendatadb), parameter **cx107** = gendatadb(ident = "NC13", formula = "C13H28", name = "N-TRIDECANE", mw = 184.3700, Tc = 675.0000, Pc = 1680000.00, Zc = 0.246000, acf = 0.618000, Tb = 508.6300, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.243240, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp109** = cpdata(cid = "NC13", ref = "Default", bib_ref = "", cptype = 8, cp = (/1.↩ 85460000e+01,-7.63600000e-03,3.46040000e-04,-4.59780000e-07,1.85090000e-10, 0.00000000e+00,0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -20.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu185** = alphadatadb(eosid="PR", cid="NC13", ref="tcPR", coeff=(/4.↩ 48200000e-01, 8.03900000e-01, 2.23430000e+00/))
- type(cidatadb), parameter **c187** = cidatadb(eosid="PR", cid="NC13", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=3.00534000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu186** = alphadatadb(eosid="SRK", cid="NC13", ref="tcRK", coeff=(/4.↩ 47100000e-01, 8.17500000e-01, 2.60970000e+00/))
- type(cidatadb), parameter **c188** = cidatadb(eosid="SRK", cid="NC13", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=7.83925000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx108** = gendatadb(ident = "NC11", formula = "C11H24", name = "N-UNDECANE", mw = 156.3120, Tc = 639.0000, Pc = 1980000.00, Zc = 0.257000, acf = 0.537000, Tb = 469.1000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/1.60541000e+01, 3.61407000e+03, -8.54500000e+01/), Tantmin = 348.0000, Tantmax = 498.0000, Zra = 0.249900, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp110** = cpdata(cid = "NC11", ref = "Default", bib_ref = "", cptype = 2, cp = (/6.52905640e+01,-9.98270000e-02,3.47249500e-03,-1.35433600e-06,2.64721000e-10, -1.45574000e-14,3.40795900e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -20.0000, Tcpmax = 1200.0000)
- type(alphadatadb), parameter **twu187** = alphadatadb(eosid="PR", cid="NC11", ref="tcPR", coeff=(/4.↩ 18500000e-01, 8.07100000e-01, 2.12240000e+00/))
- type(cidatadb), parameter **c189** = cidatadb(eosid="PR", cid="NC11", ref="tcPR", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=1.81817000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(alphadatadb), parameter **twu188** = alphadatadb(eosid="SRK", cid="NC11", ref="tcRK", coeff=(/4.↩ 01800000e-01, 8.24500000e-01, 2.58800000e+00/))
- type(cidatadb), parameter **c190** = cidatadb(eosid="SRK", cid="NC11", ref="tcRK", bib_ref="10.1016/j.fluid.↩ 2016.09.003", ciA=5.80492000e-05, ciB=0.00000000e+00, ciC=0.00000000e+00, c_type=1)
- type(gendatadb), parameter **cx109** = gendatadb(ident = "PSEUDO", formula = "XXX", name = "PSEUDO", mw = 100.0000, Tc = 100.0000, Pc = 10000000.00, Zc = 0.300000, acf = 0.000000, Tb = 0.0000, Ttr = 0.0000, Ptr = 0.0000, sref = 0.0000, href = 0.0000, DfH = 0.0000, DfG = 0.0000, psatcode = 1, ant = (/0.↩ 00000000e+00, 0.00000000e+00, 0.00000000e+00/), Tantmin = 0.0000, Tantmax = 0.0000, Zra = 0.254400, mu_dipole = 0.000000, q_quadrupole = 0.000000)
- type(cpdata), parameter **cp111** = cpdata(cid = "PSEUDO", ref = "Default", bib_ref = "", cptype = 2, cp = (/0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00, 0.↩ 00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00/), Tcpmin = -175.↩ 0000, Tcpmax = 1200.0000)
- integer, parameter **maxncdb** =109
- type(gendatadb), dimension(maxncdb), parameter **compdb** = (/ cx1,cx2,cx3,cx4,cx5, cx6,cx7,cx8,cx9,cx10, cx11,cx12,cx13,cx14,cx15, cx16,cx17,cx18,cx19,cx20, cx21,cx22,cx23,cx24,cx25, cx26,cx27,cx28,cx29,cx30, cx31,cx32,cx33,cx34,cx35, cx36,cx37,cx38,cx39,cx40, cx41,cx42,cx43,cx44,cx45, cx46,cx47,cx48,cx49,cx50, cx51,cx52,cx53,cx54,cx55, cx56,cx57,cx58,cx59,cx60, cx61,cx62,cx63,cx64,cx65, cx66,cx67,cx68,cx69,cx70, cx71,cx72,cx73,cx74,cx75, cx76,cx77,cx78,cx79,cx80, cx81,cx82,cx83,cx84,cx85, cx86,cx87,cx88,cx89,cx90, cx91,cx92,cx93,cx94,cx95, cx96,cx97,cx98,cx99,cx100, cx101,cx102,cx103,cx104,cx105, cx106,cx107,cx108,cx109 /)
- integer, parameter **maxcpdb** =111
- type(cpdata), dimension(maxcpdb), parameter **cpdb** = (/ cp1,cp2,cp3,cp4,cp5, cp6,cp7,cp8,cp9,cp10, cp11,cp12,cp13,cp14,cp15, cp16,cp17,cp18,cp19,cp20, cp21,cp22,cp23,cp24,cp25, cp26,cp27,cp28,cp29,cp30, cp31,cp32,cp33,cp34,cp35, cp36,cp37,cp38,cp39,cp40, cp41,cp42,cp43,cp44,cp45, cp46,cp47,cp48,cp49,cp50, cp51,cp52,cp53,cp54,cp55, cp56,cp57,cp58,cp59,cp60, cp61,cp62,cp63,cp64,cp65, cp66,cp67,cp68,cp69,cp70, cp71,cp72,cp73,cp74,cp75, cp76,cp77,cp78,cp79,cp80, cp81,cp82,cp83,cp84,cp85, cp86,cp87,cp88,cp89,cp90,

cp91,cp92,cp93,cp94,cp95, cp96,cp97,cp98,cp99,cp100, cp101,cp102,cp103,cp104,cp105, cp106,cp107,cp108,cp109,cp110, cp111 /)

- integer, parameter **maxtwudb** =188
- type(alphadatadb), dimension(maxtwudb), parameter **alphatwudb** = (/ twu1,twu2,twu3,twu4,twu5, twu6,twu7,twu8,twu9,twu10, twu11,twu12,twu13,twu14,twu15, twu16,twu17,twu18,twu19,twu20, twu21,twu22,twu23,twu24,twu twu26,twu27,twu28,twu29,twu30, twu31,twu32,twu33,twu34,twu35, twu36,twu37,twu38,twu39,twu40, twu41,twu42,twu43,twu44,twu45, twu46,twu47,twu48,twu49,twu50, twu51,twu52,twu53,twu54,twu55, twu56,twu57,twu58,twu59,twu60, twu61,twu62,twu63,twu64,twu65, twu66,twu67,twu68,twu69,twu70, twu71,twu72,twu73,twu74,twu75, twu76,twu77,twu78,twu79,twu80, twu81,twu82,twu83,twu84,twu85, twu86,twu87,twu88,twu89,twu90, twu91,twu92,twu93,twu94,twu95, twu96,twu97,twu98,twu99,twu100, twu101,twu102,twu103,twu104,twu105, twu106,twu107,twu108,twu109,twu110, twu111,twu112,twu113,twu114,twu115, twu116,twu117,twu118,twu119,twu120, twu121,twu122,twu123,twu124,twu125, twu126,twu127,twu128,twu129,twu130, twu131,twu132,twu133,twu134,twu135, twu136,twu137,twu138,twu139,twu140, twu141,twu142,twu143,twu144,twu145, twu146,twu147,twu148,twu149,twu150, twu151,twu152,twu153,twu154,twu155, twu156,twu157,twu158,twu159,twu160, twu161,twu162,twu163,twu164,twu165, twu166,twu167,twu168,twu169,twu170, twu171,twu172,twu173,twu174,twu175, twu176,twu177,twu178,twu179,twu180, twu181,twu182,twu183,twu184,twu185, twu186,twu187,twu188 /)
- integer, parameter **maxmcdb** =65
- type(alphadatadb), dimension(maxmcdb), parameter **alphamcdb** = (/ mc1,mc2,mc3,mc4,mc5, mc6,mc7,mc8,mc9,mc10, mc11,mc12,mc13,mc14,mc15, mc16,mc17,mc18,mc19,mc20, mc21,mc22,mc23,mc24,mc25, mc26,mc27,mc28,mc29,mc30, mc31,mc32,mc33,mc34,mc35, mc36,mc37,mc38,mc39,mc40, mc41,mc42,mc43,mc44,mc45, mc46,mc47,mc48,mc49,mc50, mc51,mc52,mc53,mc54,mc55, mc56,mc57,mc58,mc59,mc60, mc61,mc62,mc63,mc64,mc65 /)
- integer, parameter **maxcidb** =190
- type(cidatadb), dimension(maxcidb), parameter **cidb** = (/ c1,c2,c3,c4,c5, c6,c7,c8,c9,c10, c11,c12,c13,c14,c15, c16,c17,c18,c19,c20, c21,c22,c23,c24,c25, c26,c27,c28,c29,c30, c31,c32,c33,c34,c35, c36,c37,c38,c39,c40, c41,c42,c43,c44,c45, c46,c47,c48,c49,c50, c51,c52,c53,c54,c55, c56,c57,c58,c59,c60, c61,c62,c63,c64,c65, c66,c67,c68,c69,c70, c71,c72,c73,c74,c75, c76,c77,c78,c79,c80, c81,c82,c83,c84,c85, c86,c87,c88,c89,c90, c91,c92,c93,c94,c95, c96,c97,c98,c99,c100, c101,c102,c103,c104,c105, c106,c107,c108,c109,c110, c111,c112,c113,c114,c115, c116,c117,c118,c119,c120, c121,c122,c123,c124,c125, c126,c127,c128,c129,c130, c131,c132,c133,c134,c135, c136,c137,c138,c139,c140, c141,c142,c143,c144,c145, c146,c147,c148,c149,c150, c151,c152,c153,c154,c155, c156,c157,c158,c159,c160, c161,c162,c163,c164,c165, c166,c167,c168,c169,c170, c171,c172,c173,c174,c175, c176,c177,c178,c179,c180, c181,c182,c183,c184,c185, c186,c187,c188,c189,c190 /)
- integer, parameter **ncpamodels** =21
- type(cpadata), dimension(ncpamodels), parameter **cpaarray** = (/ cpa1,cpa2,cpa3,cpa4,cpa5, cpa6,cpa7,cpa8,cpa9,cpa10, cpa11,cpa12,cpa13,cpa14,cpa15, cpa16,cpa17,cpa18,cpa19,cpa20, cpa21 /)

5.10.1 Detailed Description

Automatically generated to file compdatadb.f90 using utility python code pyUtils Time stamp: 2023-09-28T12:56:50.126803.

5.11 complexmodelinit Module Reference

Initialization of complex models. These models are typically comprised of several specific sub-models, and when used together they define a known model.

Functions/Subroutines

- subroutine, public init_vtpr (ncomp, comp_string, nphases, kij_ref, alpha_ref)

  Initialize VTPR unifac-pr based model See Schmid 2014 (10.1021/ie404118f) or later.
- subroutine, public init_umr (ncomp, comp_string, nphases, kij_ref, alpha_ref)

  Initialize UMR unifac-pr based model See: 10.1021/ie049580p.

5.11.1 Detailed Description

Initialization of complex models. These models are typically comprised of several specific sub-models, and when used together they define a known model.

Author

MH, 2016-12

5.11.2 Function/Subroutine Documentation

5.11.2.1 init_umr()

```
subroutine, public complexmodelinit::init_umr (
             integer, intent(in) ncomp,
             character(len=*), intent(in) comp_string,
             integer, intent(in) nphases,
             character(len=*), intent(in), optional kij_ref,
             character(len=*), intent(in), optional alpha_ref )
```

Initialize UMR unifac-pr based model See: 10.1021/ie049580p.

Author

MH, 2016-12

Parameters

in	*ncomp*	Number of components
in	*comp_string*	String defining components. Comma or white-space separated.
in	*nphases*	Number of phases
in	*alpha_ref*	Data set numbers

5.11.2.2 init_vtpr()

```
subroutine, public complexmodelinit::init_vtpr (
             integer, intent(in) ncomp,
             character(len=*), intent(in) comp_string,
             integer, intent(in) nphases,
             character(len=*), intent(in), optional kij_ref,
             character(len=*), intent(in), optional alpha_ref )
```

Initialize VTPR unifac-pr based model See Schmid 2014 (10.1021/ie404118f) or later.

Author

MH, 2016-12

Parameters

in	*ncomp*	Number of components
in	*comp_string*	String defining components. Comma or white-space separated.
in	*nphases*	Number of phases
in	*alpha_ref*	Data set numbers

5.12 cpa_parameters Module Reference

Module for CPA parameters.

Functions/Subroutines

- integer function **getcpadataidx** (eos, compname, param_ref)

  Get the index in the CPAarray of the component having uid given by compName. idx=0 if component isn't in database.
- logical function **mixhasselfassociatingcomp** (nc, eos, complist, ref)

  Get information on if it is safe to initialize CPA ie. are there any self-associating components in the mix.
- subroutine **getcpapureparams_allcomps** (nc, comp, eosidx, ref, found, a0, b, alphaparams, eps, beta, alphacorridx, scheme)
- subroutine **getcpapureparams_singlecomp** (compname, eos, ref, found, a0, b, alphaparams, eps, beta, alphacorridx, scheme)
- subroutine **getcpakijandcombrules_allcomps** (nc, comp, eosidx, aepsbeta_kij, epsbeta_combrules)
- subroutine **getcpakij_epsbeta** (eosidx, uid1, uid2, param_ref, found, epsbetacombrules, kijepsbeta)

  Retrieve association binary interaction parameter for components uid1 and uid2. Found is true if and only if the parameters is in the database. As of now this function sets interaction parameters to 0.0 if epsBetaCombRules is not exactly what is inputted.
- real function **getcpakij_a** (eosidx, uid1, uid2, found)

  Retrieve cubic binary interaction parameter for components uid1 and uid2, with set number setno. found is true if and only if the parameter is in the database.
- subroutine **getcpageij** (mge, eosid, ref, uid1, uid2, indxi, indxj, found)

5.12.1 Detailed Description

Module for CPA parameters.

5.13 critical Module Reference

Calculate critical point of a mixture. Good initial values assumed. Based on: M.L. Michelsen, Calculation of critical points and phase boundaries in the critical region. Fluid phase equilibria, 16, 1984, pp. 57-76.

Functions/Subroutines

- subroutine, public calccritical (t, p, z, phase, ierr, tol)

  Calculate critical point for mixture given good inital guess.
- real function, public calcstabmineig (t, p, z, phase)

  Calculate minimum eigenvalue for stability matrix.
- real function, public calcstabmineigtv (t, v, z)

  Calculate minimum eigenvalue for stability matrix.
- subroutine, public calcbmatrixtv (t, v, z, zs, b, u, lambdamin, lnfugz, lnfugtz, lnfugvz)

  Calculate stability matrix (B) and minimum eigenvalue with eigenvector.
- subroutine, public stabfuntv (fun, x, param)

  Function value for calculating stability limit given v.
- subroutine, public stabjactv (df, x, param)

  Differentials of b wrpt. T.
- subroutine, public stabfun (fun, x, param)

  Function value for calculating stability limit given pressure.
- subroutine, public calccriticaltv (t, v, z, ierr, tol, v_min, p)

  Calculate critical point in variables T and V Method of Michelsen 1984 is implemented using temperature (T) and volume (V) as variables If the initial temperature T $<$ 0.0, the pseudo-critical temperature is used as initial value. If the initial volume V $<$ 0.0, the V = 4.0$*$b is used as initial value.
- subroutine, public calccriticalz (t, v, p, z, s, ierr, tol, free_comp, iter)

  Calculate critical point specified z (s=1), T (s=2), V (s=3) or P (s=4) Good initial values are assumed.
- subroutine, public critzsensitivity (z, ic, x, dxds, s, ierr)

  Sensitivities of critical equation system.
- subroutine, public calccriticalendpoint (t, vc, zc, y, vy, ierr, tol, free_comp)

  Calculate critical point in equilibrium with incipient phase Good initial values are assumed.

5.13.1 Detailed Description

Calculate critical point of a mixture. Good initial values assumed. Based on: M.L. Michelsen, Calculation of critical points and phase boundaries in the critical region. Fluid phase equilibria, 16, 1984, pp. 57-76.

5.13.2 Function/Subroutine Documentation

5.13.2.1 calcbmatrixtv()

```
subroutine, public critical::calcbmatrixtv (
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(nc), intent(in) z,
             real, dimension(nc), intent(in) zs,
             real, dimension(nc,nc), intent(out) b,
             real, dimension(nc), intent(out) u,
             real, intent(out) lambdamin,
             real, dimension(nc), intent(out) lnfugz,
             real, dimension(nc), intent(out) lnfugtz,
             real, dimension(nc), intent(out) lnfugvz )
```

Calculate stability matrix (B) and minimum eigenvalue with eigenvector.

Author

MH, 2015-11

Parameters

in	*z*	Overall composition
in	*zs*	z
in	*t*	Temperature
in	*v*	Specific volume (m3/mol)
out	*b*	Stability matrix
out	*lambdamin*	Eigenvalue
out	*u*	Eigenvector

5.13.2.2 calccritical()

```
subroutine, public critical::calccritical (
             real, intent(inout) t,
             real, intent(inout) p,
             real, dimension(nc), intent(in) z,
             integer, intent(in) phase,
             integer, intent(out) ierr,
             real, intent(in), optional tol )
```

Calculate critical point for mixture given good inital guess.

Todo Add v=sqrt(z) to paramater vector

Author

MH, 2014-11

Parameters

in	*z*	Trial composition (Overall compozition)
in,out	*t*	Temperature [K]
in,out	*p*	Pressure [Pa]
in	*phase*	Phase identifer
out	*ierr*	Error flag
in	*tol*	Toleranse

5.13.2.3 calccriticalendpoint()

```
subroutine, public critical::calccriticalendpoint (
             real, intent(inout) t,
             real, intent(inout) vc,
             real, dimension(nc), intent(inout) zc,
             real, dimension(nc), intent(inout) y,
             real, intent(inout) vy,
             integer, intent(out) ierr,
```

```
            real, intent(in), optional tol,
            integer, intent(in), optional free_comp )
```

Calculate critical point in equilibrium with incipient phase Good initial values are assumed.

Author

MH, 2019-04

Parameters

in,out	*zc*	Trial composition (Overall compozition)
in,out	*t*	Temperature [K]
in,out	*vc*	Volume [m3/mol]
in,out	*vy*	Volume [m3/mol]
in,out	*y*	Incipient phase
out	*ierr*	Error flag
in	*tol*	Toleranse
in	*free_comp*	Component variable

5.13.2.4 calccriticaltv()

```
subroutine, public critical::calccriticaltv (
            real, intent(inout) t,
            real, intent(inout) v,
            real, dimension(nc), intent(in) z,
            integer, intent(out) ierr,
            real, intent(in), optional tol,
            real, intent(in), optional v_min,
            real, intent(out), optional p )
```

Calculate critical point in variables T and V Method of Michelsen 1984 is implemented using temperature (T) and volume (V) as variables If the initial temperature T $<$ 0.0, the pseudo-critical temperature is used as initial value. If the initial volume V $<$ 0.0, the V = 4.0$*$b is used as initial value.

Author

MH, 2016-01

Parameters

in	*z*	Trial composition (Overall compozition)
in,out	*t*	Temperature [K]
in,out	*v*	Pressure [m3/mol]
out	*ierr*	Error flag
in	*tol*	Toleranse
in	*v_min*	Override lower volume limit (m3/mol)
out	*p*	Pressure (Pa)

5.13.2.5 calccriticalz()

```
subroutine, public critical::calccriticalz (
             real, intent(inout) t,
             real, intent(inout) v,
             real, intent(inout) p,
             real, dimension(nc), intent(inout) z,
             integer, intent(in) s,
             integer, intent(out) ierr,
             real, intent(in), optional tol,
             integer, intent(in), optional free_comp,
             integer, intent(out), optional iter )
```

Calculate critical point specified z (s=1), T (s=2), V (s=3) or P (s=4) Good initial values are assumed.

Author

MH, 2019-04

Parameters

in,out	*z*	Trial composition (Overall compozition)
in,out	*t*	Temperature [K]
in,out	*v*	Volume [m3/mol]
in,out	*p*	Pressure [Pa]
in	*s*	Specification
out	*ierr*	Error flag
in	*tol*	Toleranse
in	*free_comp*	Component variable
out	*iter*	Number of iterations

5.13.2.6 calcstabmineig()

```
real function, public critical::calcstabmineig (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             integer, intent(in) phase )
```

Calculate minimum eigenvalue for stability matrix.

Author

MH, 2013-10-10

Parameters

in	*z*	Overall composition
in	*t*	Temperature
in	*p*	Pressure
in	*phase*	Phase identifer

Returns

Eigenvalue

5.13.2.7 calcstabmineigtv()

```
real function, public critical::calcstabmineigtv (
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(nc), intent(in) z )
```

Calculate minimum eigenvalue for stability matrix.

Author

MH, 2015-11

Parameters

in	*z*	Overall composition
in	*t*	Temperature (K)
in	*v*	Volume (m3/mol)

Returns

Eigenvalue

5.13.2.8 critzsensitivity()

```
subroutine, public critical::critzsensitivity (
             real, dimension(nc), intent(in) z,
             integer, intent(in) ic,
             real, dimension(4), intent(in) x,
             real, dimension(4), intent(out) dxds,
             integer, intent(in) s,
             integer, intent(out) ierr )
```

Sensitivities of critical equation system.

Author

MH, 2019-04

Parameters

out	*dxds*	System sensitivities
in	*x*	Variables
in	*z*	Composition
in	*s*	Specification
in	*ic*	Component specification
out	*ierr*	Error flag

5.13.2.9 stabfun()

```
subroutine, public critical::stabfun (
             real, dimension(1), intent(out) fun,
             real, dimension(1), intent(in) x,
             real, dimension(nc+2), intent(in) param )
```

Function value for calculating stability limit given pressure.

Author

MH, 2015-11

Parameters

out	*fun*	Function value
in	*x*	Variables
in	*param*	Parameters

5.13.2.10 stabfuntv()

```
subroutine, public critical::stabfuntv (
             real, dimension(1), intent(out) fun,
             real, dimension(1), intent(in) x,
             real, dimension(nc+1), intent(in) param )
```

Function value for calculating stability limit given v.

Author

MH, 2015-11

Parameters

out	*fun*	Function value
in	*x*	Variables
in	*param*	Parameters

5.13.2.11 stabjactv()

```
subroutine, public critical::stabjactv (
             real, dimension(1,1), intent(out) df,
             real, dimension(1), intent(in) x,
             real, dimension(nc+1), intent(in) param )
```

Differentials of b wrpt. T.

Author

MH, 2015-11

Parameters

out	*df*	Function differential
in	*x*	Variables
in	*param*	Parameters

5.14 cubic Module Reference

This module contains methods for various cubic equation of states The cubic eos's are formulated using the m1 and the m2. Suppoerted cubic EOS's SRK - Soave-Redlich-Kwong PR - Peng Robinson VdW - Van Der Waals RK - Redlich-Kwong SW - Schmidt and Wenzel PT - Patel-Teja.

Data Types

- type cbbig

Functions/Subroutines

- subroutine **cbcalcderivatives** (nc, cbeos, t, p, zfac)
- subroutine, public cbcalcderivatives_svol (nc, cbeos, t, v)
- subroutine, public **cbcalcpressure** (nc, cbeos, t, v, z, p, dpdv, dpdt, d2pdv2, dpdz, recalculate)

  Explicit calculation of pressure including the volume and temperature derivative.
- subroutine cb_solve_cubic_zfac (p2, p1, p0, z, lconverged)

  Find a root z0 of the cubic polynomial.
- subroutine cb_cubic_second_zfac (p2, p1, z0, zl, zg, iflag)

  Having found the compressibility factor z = z0, that satisfies f(z0) = 0 for the cubic polinomial.
- subroutine cbsolvecubiczfac (pp, qq, rr, z, ifail)

  This subroutine solves a cubic polynomial.
- subroutine **cbsolvecubiczfacminimumgibb** (cbeos, t, p, pp, qq, rr, big, zfac, ifail, phase)
- subroutine, public **cbcalczfac** (nc, cbeos, t, p, z, iphase, zfac, gflag_opt, dzdt, dzdp, dzdz, mingphase)
- subroutine **cbcalczfacdiff** (nc, cbeos, t, p, zfac, dzdt, dzdp, dzdz)
- subroutine, public cbgres (cbeos, t, p, zfac, gr, dgrdt, dgrdp)

  The function to find the residual Gibbs free energy for the cubic equation of state.
- subroutine, public cbcalcfug (nc, cbeos, t, p, zfac, res_fug, dlnfdt, dlnfdp, dlnfdz)

  The subroutine finds the fugacity coefficients and derivatives.
- subroutine, public cbcalcentropy (nc, cbeos, t, p, z, phase, res_entropy, gflag_opt, dsdt, dsdp, dsdz)

  The subroutine finds the entropy residual of cubic Equations of State optionally the derivatives.
- subroutine, public cbcalcenthalpy (nc, cbeos, t, p, z, phase, res_enthalpy, gflag_opt, dhdt, dhdp, dhdz)

  The subroutine finds the enthalpy residual of cubic Equations of State optionally the derivatives.
- subroutine cbcalcinnerenergy (nc, cbeos, t, v, z, u, dudt, dudv, recalculate)

  The subroutine finds the innerenergy residual of cubic Equations of State optionally the derivatives.
- subroutine, public cbcalcpseudo (nc, cbeos, z, tpc, ppc, zpc, vpc)

  The subroutine calculates the pseudocritical temperature, pressure, acentric factor compressibility and volume as function of composition. The pseudo-critical point is the state where with constant Temperature: (dP/dV)=(d2P/d↩ V^2)=0.
- subroutine, public **cbdumpeosdata** (nc, cbeos, t, p, zfac)
- subroutine, public cbcalcfreeenergy (nc, cbeos, t, v, z, y, dydt, dydv, recalculate)

  Subroutine finds Helmholtz free energy residual of cubic EoS optionally the derivatives with respect to T, v.
- subroutine, public cbcalcfres (nc, cbeos, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_vv, f_tn, f_vn, f_nn, recalculate)

  Calculate resudial reduced Helmholtz and differentials.
- subroutine **calccbfder_res_si** (nc, cbeos, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_vv, f_tn, f_vn, f_nn, f_vvv, recalculate)

  Calculates the contibution to the reduced residual Helmholtz energy F coming from the cubic eos, along with its derivatives. Note!!! Input and output in SI units.

5.14.1 Detailed Description

This module contains methods for various cubic equation of states The cubic eos's are formulated using the m1 and the m2. Suppoerted cubic EOS's SRK - Soave-Redlich-Kwong PR - Peng Robinson VdW - Van Der Waals RK - Redlich-Kwong SW - Schmidt and Wenzel PT - Patel-Teja.

5.14.2 Function/Subroutine Documentation

5.14.2.1 cb_cubic_second_zfac()

```
subroutine cubic::cb_cubic_second_zfac (
             real, intent(in) p2,
             real, intent(in) p1,
             real, intent(in) z0,
             real, intent(out) zl,
             real, intent(out) zg,
             integer, intent(out) iflag )
```

Having found the compressibility factor z = z0, that satisfies f(z0) = 0 for the cubic polinomial.

$$f(z) = z^3 + p2z^2 + p1z + p0,$$

we can find the others analytically by solving the quadratic equation

$$\begin{aligned} g(z) &= z^2 + (p2 + z0)z + (p2z0 + z0^2 + p1) \\ &= z^2 + q1z + q0 \\ &= 0. \end{aligned}$$

Author

MAG, 2013-09

Parameters

p2	- Polynomial coefficient
p1	- Polynomial coefficient
z0	- Root of polynomial

Returns

zl - Smallest real root found

zg - Largest real root found

iflag - Return code ifalg = 1 : Converged to a single, real root iflag = 2 : Converged to three, real though maybe degenerate, roots

5.14.2.2 cb_solve_cubic_zfac()

```
subroutine cubic::cb_solve_cubic_zfac (
             real, intent(in) p2,
             real, intent(in) p1,
             real, intent(in) p0,
             real, intent(inout) z,
             logical, intent(out) lconverged )
```

Find a root z0 of the cubic polynomial.

$$f(z) = z^3 + p2z^2 + p1z + p0,$$

to solve for for the compressibility factor.

Author

MAG, 2013-09

Parameters

p2	- Polynomial coefficient
p1	- Polynomial coefficient
p0	- Polynomial coefficient

Returns

z - Root found, initial guess as input

lconverged - True if solver converged

5.14.2.3 cbcalcderivatives_svol()

```
subroutine, public cubic::cbcalcderivatives_svol (
             integer, intent(in) nc,
             class(cb_eos), intent(inout) cbeos,
             real, intent(in) t,
             real, intent(in) v )
```

For three param like Patel-Teja, m1 = m1(sumb,sumc) nad m2 = m2(sumb,sumc) For Schmidt and Wenzel, m1 = m1(sumc) and m2 = m2(sumc) - or m2(acfmix)

Define reduced Helmholz energy: as F = -g - (amix(T)/T) $*$ f

n = v - cbeossumb n1 = v - cbeosm1 $*$ cbeossumb n2 = v - cbeosm2 $*$ cbeossumb den = n1$*$n2 lnn = log(n2/n1) or log n2 - log n1 h = lnn
Michelsen | Thermopack

5.14.2.4 p88-91 |

D | cbeossuma, Di | cbeosa(i) Dj | cbeosa(j) Dt | cbeosat Dtt | cbeosatt Dij | cbeosaij(i,j) Dit | cbeosait(i) B | cbeossumb, Bi | cbeosb(i),

5.14.2.5 Bij | cbeos%bij = 0

5.14.2.6 cbcalcenthalpy()

```
subroutine, public cubic::cbcalcenthalpy (
             integer, intent(in) nc,
             class(cb_eos), intent(inout) cbeos,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             integer, intent(in) phase,
             real, intent(out) res_enthalpy,
             integer, intent(in) gflag_opt,
             real, intent(out), optional dhdt,
             real, intent(out), optional dhdp,
             real, dimension(nc), intent(out), optional dhdz )
```

The subroutine finds the enthalpy residual of cubic Equations of State optionally the derivatives.

Parameters

T	- Temperature [K]
P	- Pressure [Pa]
Z	- Composition [-]
phase	- Phase (1=liquid, 2=vapour)
gflag_opt	The TPlib "Gunder" flag 1: Normal 2: if the metastable maxima or minima of the z-factor are to be returned 3: If possibilities of having three roots, return the one having minimum Gibbs free energy is to be returned - calls cbGres
res_enthalpy	- Residual enthalpy
dhdt	- Residual enthalpy differential wrpt. temperature
dhdp	- Residual enthalpy differential wrpt. pressure
dhdz	- Residual enthalpy differential wrpt. mole numbers

Author

Oivind Wilhelmsen

Date

2012-06-14

5.14.2.7 cbcalcentropy()

```
subroutine, public cubic::cbcalcentropy (
             integer, intent(in) nc,
             class(cb_eos), intent(inout) cbeos,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             integer, intent(in) phase,
             real, intent(out) res_entropy,
             integer, intent(in) gflag_opt,
             real, intent(out), optional dsdt,
             real, intent(out), optional dsdp,
             real, dimension(nc), intent(out), optional dsdz )
```

The subroutine finds the entropy residual of cubic Equations of State optionally the derivatives.

Parameters

T	- Temperature [K]
P	- Pressure [Pa]
Z	- Composition [-]
phase	- Phase (1=liquid, 2=vapour)
gflag_opt	The TPlib "Gunder" flag 1: Normal 2: if the metastable maxima or minima of the z-factor are to be returned 3: If possibilities of having three roots, return the one having minimum Gibbs free energy is to be returned - calls cbGres
res_entropy	- Residual entropy
dsdt	- Residual entropy differential wrpt. temperature
dsdp	- Residual entropy differential wrpt. pressure
dsdz	- Residual entropy differential wrpt. mole numbers

Author

Oivind Wilhelmsen

Date

2012-06-13

5.14.2.8 cbcalcfreeenergy()

```
subroutine, public cubic::cbcalcfreeenergy (
             integer, intent(in) nc,
             class(cb_eos), intent(inout) cbeos,
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(1:nc), intent(in) z,
             real, intent(out) y,
             real, intent(out), optional dydt,
             real, intent(out), optional dydv,
             logical, intent(in), optional recalculate )
```

Subroutine finds Helmholtz free energy residual of cubic EoS optionally the derivatives with respect to T, v.

Parameters

T	Temperature [K]
v	Specific volume [m3/kmol]
Z	Mole fraction [-]
dYdt	Temperature derivative [J/kmolK]
dYdv	Pressure derivative [J/m3] 0: No derivatives 1: Analytical derivatives

Return values

Y	Free energy with derivatives [-]

Author

GL, 2015-01-22

5.14.2.9 cbcalcfres()

```
subroutine, public cubic::cbcalcfres (
             integer, intent(in) nc,
             class(cb_eos), intent(inout) cbeos,
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(nc), intent(in) n,
             real, intent(out), optional f,
             real, intent(out), optional f_t,
             real, intent(out), optional f_v,
             real, dimension(nc), intent(out), optional f_n,
             real, intent(out), optional f_tt,
             real, intent(out), optional f_tv,
             real, intent(out), optional f_vv,
             real, dimension(nc), intent(out), optional f_tn,
             real, dimension(nc), intent(out), optional f_vn,
             real, dimension(nc,nc), intent(out), optional f_nn,
             logical, intent(in), optional recalculate )
```

Calculate resudial reduced Helmholtz and differentials.

Parameters

T	- Temperature [K]
v	- Specific volume [m3/kmol]
n	- Mole numbers [mol]
F...	- Resudial reduced Helmholtz differentiuals

Author

Morten Hammer

Date

2015-09

5.14.2.10 cbcalcfug()

```
subroutine, public cubic::cbcalcfug (
             integer, intent(in) nc,
             class(cb_eos), intent(inout) cbeos,
             real, intent(in) t,
             real, intent(in) p,
             real, intent(in) zfac,
             real, dimension(nc), intent(out) res_fug,
             real, dimension(nc), intent(out), optional dlnfdt,
             real, dimension(nc), intent(out), optional dlnfdp,
             real, dimension(nc,nc), intent(out), optional dlnfdz )
```

The subroutine finds the fugacity coefficients and derivatives.

Parameters

T	- Temperature [K]
P	- Pressure [Pa]

Parameters

Zfac	- Compressibility factor
dlnfdt	- Fugacity coefficients differential wrpt. temperature
dlnfdp	- Fugacity coefficients differential wrpt. pressure
dlnfdz	- Fugacity coefficients differential wrpt. mole numbers
res_fug	- Fugacity coefficients

Author

Oivind Wilhelmsen

Date

2012-06-13

5.14.2.11 cbcalcinnerenergy()

```
subroutine cubic::cbcalcinnerenergy (
             integer, intent(in) nc,
             class(cb_eos), intent(inout) cbeos,
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(1:nc), intent(in) z,
             real, intent(out) u,
             real, intent(out), optional dudt,
             real, intent(out), optional dudv,
             logical, intent(in), optional recalculate )
```

The subroutine finds the innerenergy residual of cubic Equations of State optionally the derivatives.

Parameters

T	The temperature [K]
v	The specific volume [m3/kmole]
Z	The mole fraction [-]
dudt	Temperature derivative [J/kmoleK]
dudv	Pressure derivative [J/m3] 0: No derivatives 1: Analytical derivatives

Return values

energy	The internal energy with derivatives [-]

Author

Morten Hammer

5.14.2.12 cbcalcpseudo()

```
subroutine, public cubic::cbcalcpseudo (
             integer, intent(in) nc,
             class(cb_eos), intent(inout) cbeos,
             real, dimension(1:nc), intent(in) z,
             real, intent(out) tpc,
             real, intent(out) ppc,
```

```
            real, intent(out) zpc,
            real, intent(out) vpc )
```

The subroutine calculates the pseudocritical temperature, pressure, acentric factor compressibility and volume as function of composition. The pseudo-critical point is the state where with constant Temperature: (dP/dV)=(d2P/d↩ V^2)=0.

Parameters

Z	- Composition [-]

Returns

Tpc - Pseudo critical temperature [K]

Ppc - Pseudo critical pressure [Pa]

Zpc - Pseudo critical compressibility [-]

Vpc - Pseudo critical volume [m^3/kmole]

Author

Oivind Wilhelmsen

Date

2012-06-14

For the Schmidt and Wenzel EOS

5.14.2.13 cbgres()

```
subroutine, public cubic::cbgres (
            class(cb_eos), intent(inout) cbeos,
            real, intent(in) t,
            real, intent(in) p,
            real, intent(in) zfac,
            real, intent(out) gr,
            real, intent(out), optional dgrdt,
            real, intent(out), optional dgrdp )
```

The function to find the residual Gibbs free energy for the cubic equation of state.
This function is called if the "gflag" is set to 3 in the cbCalcZfac routine.

Parameters

T	- Temperature [K]
p	- Pressure [Pa]
zfac	- Compressibility factor
gr	- The residual Gibbs free energy
dgrdt	- The residual Gibbs free energy temperature differential
dgrdp	- The residual Gibbs free energy pressure differential

5.14.2.14 cbsolvecubiczfac()

```
subroutine cubic::cbsolvecubiczfac (
            real, intent(in) pp,
            real, intent(in) qq,
            real, intent(in) rr,
```

```
            real, intent(inout) z,
            integer, intent(out) ifail )
```

This subroutine solves a cubic polynomial.
f(z) = z**3 + pp*z**2 + qq*z + rr = 0
with a third order method
Relaxation is necessarry in the critical region because of both the first and second derivative approach zero
Original TPlib routine converted to use new data structure

Parameters

pp	- 1'st polynmominal coefficient
qq	- 2'nd polynominal coefficient
rr	- 3'rd polynominal coefficient
z	- Initial and return value for the compressibility factor

Returns

ifail - Return code ifail = 0 : Converged, the gradient toward the solution is positive ifail = 1 : Not converged after maximum number of iterations ifail = 2 : Converged, the gradient toward the solution is negative ifail = 3 : Not converged and the maximum number of relaxtions has been reach wi

5.15 cubic_eos Module Reference

The module eosdata contains the definitions of the equation of state, mixing rule and the interaction parameters.

Data Types

- type alpha_label_mapping
- type cb_eos
- type cpa_eos
- type cpakijdata

  Temperature-independent interaction parameters for.
- type fraction
- type intergedatadb
- type kijdatadb
- type lijdatadb
- type lk_eos
- type mix_label_mapping
- type mixexcessgibbs
- type mixwongsandler
- type singledata

Functions/Subroutines

- subroutine **ws_deallocate** (mixws)
- subroutine **ws_allocate_and_init** (mixws, nc)
- subroutine **assign_ws_mix** (mixws1, mixws2)
- subroutine **excess_gibbs_deallocate** (mixge)
- subroutine **excess_gibbs_allocate_and_init** (mixge, nc)
- subroutine **assign_excess_gibbs_mix** (mixge1, mixge2)
- type(cb_eos) function **cubic_eos_constructor** (nc, eos_label)

  Allocate memory for cubic eos.
- type(lk_eos) function **lk_eos_constructor** (nc, eos_label)

  Allocate memory for LK eos.
- type(cpa_eos) function **cpa_eos_constructor** (nc, eos_label)

Allocate memory for CPA eos.

- subroutine **assign_cubic_eos** (this, other)
- subroutine allocate_and_init_cubic_eos (eos, nc, eos_label)
- subroutine **cubic_eos_dealloc** (eos)
- logical function **ishvmixmodel** (mix_idx)
- logical function **isgemixmodel** (mix_idx)
- integer function **get_mix_db_idx** (short_label)
- integer function **get_alpha_db_idx** (short_label)
- integer function **get_alpha_db_idx_from_alpha_idx** (alpha_idx)
- integer function **eos_to_classic_alpha_db_idx** (eosidx)
- logical function **is_classic_alpha** (alphaidx)
- subroutine **get_covolumes** (b)

  Get covolumes.
- subroutine **get_energy_constants** (a)

  Get energy constant.
- real function get_b_linear_mix (z)

  Get linear combination of b_i.

Variables

- integer, parameter **ndegreepoly** =2
- integer, parameter **cbmixclassicgroup** = 1

  Classic kij type mixing.
- integer, parameter **cbmixvdw** = 11

  Classic vdW mixing rule for am and bm - using k_ij == k_ji.
- integer, parameter **cbmixvdwcpa** = 12

  CPA mixing rule (same as cbMixVdW, but kij from another db)
- integer, parameter **cbmixreid** = 13

  Unsymmertic mixing rule where k_ij can be different than k_ji.
- integer, parameter **cbmixgegroup** = 2
- integer, parameter **cbmixhuronvidal** = 21

  Huron vidal mixing rule.
- integer, parameter **cbmixhuronvidal2** = 22

  Huron vidal mixing rule.
- integer, parameter **cbmixnrtl** = 23

  NRTL mixing rule.
- integer, parameter **cbmixunifac** = 24

  UNIFAC mixing rule.
- integer, parameter **cbmixhvcpa** = 25

  Huron Vidal mixing rule (classic, but kij from another db)
- integer, parameter **cbmixhvcpa2** = 26

  Huron Vidal mixing rule (classic, but kij from another db)
- integer, parameter **cbmixwongsandler** = 3

  Wong Sandler mixing rule.
- integer, parameter **cbmixwscpa** = 31

  Wong-Sandler mixing rule for CPA.
- integer, parameter **cbmixhvwongsandler** = 32

  Wong-Sandler mixing rule with HV formulation of NRTL.
- integer, parameter **n_mix_rules** = 12

- type(mix_label_mapping), dimension(n_mix_rules), parameter **mix_label_db** = (/ mix_label_mapping(mix_idx_group=cbMixClassicGroup, mix_idx=cbMixVdW, short_label="VDW", label="Classic", alias = "CLASSIC"), mix_label_mapping(mix_idx_group=cbMixClassicGroup, mix_idx=cbMixVdWCPA, short_label="Classic(CPA)", label="Classic(CPA)", alias = ""), mix_label_mapping(mix_idx_group=cbMixClassicGroup, mix_idx=cbMixReid, short_label="Reid", label="Reid", alias = ""), mix_label_mapping(mix_idx_group=cbMixGEGroup, mix_idx=cbMixHuronVidal, short_label="HV", label="Huron-Vidal", alias = "HV1/HV0"), mix_label_mapping(mix_idx_group=cbMixGEGroup, mix_idx=cbMixHuronVidal2, short_label="HV2", label="Huron-Vidal2", alias = ""), mix_label_mapping(mix_idx_group=cbMixWongSandler, mix_idx=cbMixWongSandler, short_label="WongSandler", label="Wong-Sandler", alias = "WS"), mix_label_mapping(mix_idx_group=cbMixGEGroup, mix_idx=cbMixNRTL, short_label="NRTL", label="NRTL", alias = ""), mix_label_mapping(mix_idx_group=cbMixGEGroup, mix_idx=cbMixUNIFAC, short_label="UNIFAC", label="UNIFAC", alias = "UMR/VTPR"), mix_label_mapping(mix_idx_group=cbMixGEGroup, mix_idx=cbMixHVCPA, short_label="HVCPA", label="HVCPA", alias = ""), mix_label_mapping(mix_idx_group=cbMixGEGroup, mix_idx=cbMixHVCPA2, short_label="HVCPA2", label="HVCPA2", alias = ""), mix_label_mapping(mix_idx_group=cbMixWongSandler, mix_idx=cbMixWSCPA, short_label="WongSandler", label="WSCPA", alias = ""), mix_label_mapping(mix_idx_group=cbMixWongSandler, mix_idx=cbMixHVWongSandler, short_label="HVWongSandler", label="HVWongSandler", alias = "") /)
- integer, parameter **nhvcorrs** = 5
- integer, dimension(nhvcorrs), parameter **hvcorrindices** = (/ cbMixHuronVidal, cbMixHuronVidal2, cbMixHVCPA, cbMixHVCPA2, cbMixHVWongSandler/)
- integer, parameter **ngecorrs** = 7
- integer, dimension(ngecorrs), parameter **gecorrindices** = (/ HVCorrIndices, cbMixNRTL, cbMixGEGroup/)
- integer, parameter **cbalphaclassicidx** = 1

  Classic alpha-correlation VdW, RK, SRK and PR.
- integer, parameter **cbalphatwuidx** = 2

  Twu-Coon-Bluck-Cunninghan exponential formulation.
- integer, parameter **cbalphamcidx** = 3

  Mathias-Copeman expression for polar substances.
- integer, parameter **cbalphagergidx** = 4

  Gerg-Water(PR) expression for polar substances Q1,Q2 and Q3.
- integer, parameter **cbalphaclassicfitidx** = 5

  Classic alpha-corr where with replaced by a fitted parameter. Used in CPA.
- integer, parameter **cbalphaumridx** = 6

  Classic alpha-corr with another function m(acf)
- integer, parameter **cbalphagbidx** = 7

  Alpha-corr of Graboski and Daubert (10.1021/i260068a009)
- integer, parameter **cbalpharkidx** = 8

  Alpha-corr of Redlich-Kwong (10.1021/cr60137a013)
- integer, parameter **cbalphasoaveidx** = 9

  Alpha-corr of Soave ()
- integer, parameter **cbalphapridx** = 10

  Alpha-corr of Peng-Robinson.
- integer, parameter **cbalphaptidx** = 11

  Alpha-corr of Patel-Teja.
- integer, parameter **cbalphaswidx** = 12

  Alpha-corr of Schmidt-Wensel.
- integer, parameter **cbalphavdwidx** = 13

  Alpha-corr of van der Waals.
- integer, parameter **cbalphapr78idx** = 14

  Peng-Robinson alpha-corr for w>0.491 (RR-28 GPA)
- integer, parameter **n_alpha_corrs** = 14

- type(alpha_label_mapping), dimension(n_alpha_corrs), parameter **alpha_corr_db** = (/ alpha_label_mapping(alpha_idx=cbAlphaClassicIdx, n_param=1, short_label="CLASSIC", description="Classic alpha-correlation VdW, SRK, PR, PT and SW", classic_for_eos_idx=-1), alpha_label_mapping(alpha_idx=cbAlphaTwuIdx, n_param=3, short_label="TWU", description="Twu-Coon-Bluck-Cunninghan exponential formulation", classic_for_eos_idx=-1), alpha_label_mapping(alpha_idx=cbAlphaMcIdx, n_param=3, short_label="MC", description="Mathias-Copeman expression for polar substances", classic_for_eos_idx=-1), alpha_label_mapping(alpha_idx=cbAlphaGergIdx, n_param=3, short_label="GERG", description="Gerg-Water(PR) expression for polar substances", classic_for_eos_idx=-1), alpha_label_mapping(alpha_idx=cbAlphaClassicFitIdx, n_param=1, short_label="CLASSICFIT", description="Classic alpha-corr with fitted parameter. Used in CPA.", classic_for_eos_idx=-1), alpha_label_mapping(alpha_idx=cbAlphaUMRIdx, n_param=1, short_label="UMR", description="Classic alpha-corr with another function m(acf)", classic_for_eos_idx=-1), alpha_label_mapping(alpha_idx=cbAlphaGBIdx, n_param=1, short_label="GD", description="Alpha-corr of Graboski and Daubert (10.1021/i260068a009)", classic_for_eos_idx=-1), alpha_label_mapping(alpha_idx=cbAlphaRKIdx, n_param=1, short_label="RK", description="Alpha-corr of Redlich-Kwong (10.1021/cr60137a013)", classic_for_eos_idx=-1), alpha_label_mapping(alpha_idx=cbAlphaSoaveIdx, n_param=1, short_label="SOAVE", description="Alpha-corr of Soave", classic_for_eos_idx=cbSRK), alpha_label_mapping(alpha_idx=cbAlphaPRIdx, n_param=1, short_label="PR", description="Alpha-corr of Peng-Robinson", classic_for_eos_idx=cbPR), alpha_label_mapping(alpha_idx=cbAlphaPTIdx, n_param=1, short_label="PT", description="Alpha-corr of Patel-Teja", classic_for_eos_idx=cbPT), alpha_label_mapping(alpha_idx=cbAlphaSWIdx, n_param=1, short_label="SW", description="Alpha-corr of cbAlphaSWIdx", classic_for_eos_idx=cbSW), alpha_label_mapping(alpha_idx=cbAlphaVDWIdx, n_param=1, short_label="VDW", description="Alpha-corr of cbAlphaVDWIdx", classic_for_eos_idx=cbVDW), alpha_label_mapping(alpha_idx=cbAlphaPR78Idx, n_param=1, short_label="PR78", description="Alpha-corr of cbAlphaPR78Idx", classic_for_eos_idx=-1) /)
- integer, parameter **nbetacorrs** = 2
- integer, parameter **cbbetaclassicidx** = 1

  Classic beta-correlation yielding the classic, temperature-independent covolume.
- integer, parameter **cbbetaquantumidx** = 2

  Beta-correlation for quantum fluids He, Ne, H2, D2.
- character(len=11), dimension(nbetacorrs), parameter **betacorrnames** = (/ "CLASSIC ", "QUANTUM " /)
- integer, dimension(nbetacorrs), parameter **betacorrnumparams** = (/0,2/)

5.15.1 Detailed Description

The module eosdata contains the definitions of the equation of state, mixing rule and the interaction parameters.

5.15.2 Function/Subroutine Documentation

5.15.2.1 allocate_and_init_cubic_eos()

```
subroutine cubic_eos::allocate_and_init_cubic_eos (
             class(cb_eos), intent(inout) eos,
             integer, intent(in) nc,
             character(len=*), intent(in) eos_label )
```

Parameters

in	*eos_label*	EOS label

5.15.2.2 get_b_linear_mix()

```
real function cubic_eos::get_b_linear_mix (
             real, dimension(nc), intent(in) z )
```

Get linear combination of b_i.

Author

MH, 2020-07

Parameters

in	*z*	Molar compozition [-]

Returns

m3/mol

5.16 eos Module Reference

Interface to thermodynamic models. Currently ThermoPack and TREND equations of state are supported.

Functions/Subroutines

- subroutine, public thermo (t, p, z, phase, lnfug, lnfugt, lnfugp, lnfugx, ophase, metaextremum, v)

  Calculate fugasity coefficient and differentials given composition, temperature and pressure.
- subroutine, public zfac (t, p, x, phase, z, dzdt, dzdp, dzdx)

  Calculate single-phase compressibility factor given composition, temperature and pressure.
- subroutine, public specificvolume (t, p, x, phase, v, dvdt, dvdp, dvdx)

  Calculate single-phase specific volume given composition, temperature and pressure.
- real function, public twophasespecificvolume (t, p, z, x, y, beta, phase, betal)

  Calculate gas-liquid or single-phase specific volume given composition, temperature and pressure.
- subroutine, public enthalpy (t, p, x, phase, h, dhdt, dhdp, dhdx, residual)

  Calculate single-phase specific enthalpy given composition, temperature and pressure.
- real function, public twophaseenthalpy (t, p, z, x, y, beta, phase, betal)

  Calculate gas-liquid or single-phase specific enthalpy given composition, temperature and pressure.
- subroutine, public entropy (t, p, x, phase, s, dsdt, dsdp, dsdx, residual)

  Calculate single-phase specific entropy given composition, temperature and pressure.
- real function, public twophaseentropy (t, p, z, x, y, beta, phase, betal)

  Calculate gas-liquid or single-phase specific entropy given composition, temperature and pressure.
- real function, public twophaseinternalenergy (t, p, z, x, y, beta, phase, betal)

  Calculate gas-liquid or single-phase internal energy given composition, temperature, pressure and a phase state given by phase/beta.
- subroutine, public pseudo (x, tpc, ppc, acfpc, zpc, vpc)

  Calculate pseudo critical point.
- subroutine, public pseudo_safe (x, tpc, ppc, zpc, vpc)

  Calculate pseudo critical point, or use estimate from alternative EoS if necessary.
- subroutine, public getcriticalparam (i, tci, pci, oi, vci, tnbi)

  Get critical state (and more) of pure fluid.
- subroutine, public residualgibbs (t, p, z, phase, gr, dgrdt, dgrdp, dgrdn, metaextremum)

  Calculate residual Gibbs energy. Unit: J/mol.
- real function, public moleweight (z)

  Get mole weight. Unit: g/mol.
- real function, public compmoleweight (j)

  Get component mole weight. Unit: g/mol.

5.16.1 Detailed Description

Interface to thermodynamic models. Currently ThermoPack and TREND equations of state are supported.

Author

MH, 2012-01-25

5.16.2 Function/Subroutine Documentation

5.16.2.1 compmoleweight()

```
real function, public eos::compmoleweight (
            integer, intent(in) j )
```

Get component mole weight. Unit: g/mol.

Author

MH, 2013-03-06

Parameters

in	*j*	Component index

Returns

g/mol - Mole weight

5.16.2.2 enthalpy()

```
subroutine, public eos::enthalpy (
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(1:nc), intent(in) x,
            integer, intent(in) phase,
            real, intent(out) h,
            real, intent(out), optional dhdt,
            real, intent(out), optional dhdp,
            real, dimension(1:nc), intent(out), optional dhdx,
            logical, intent(in), optional residual )
```

Calculate single-phase specific enthalpy given composition, temperature and pressure.

Author

MH, 2012-03-20

Parameters

in	*phase*	Phase identifyer
in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*x*	Compozition
out	*h*	J/mol - Specific enthalpy
out	*dhdt*	J/mol/K - Specific enthalpy differential wrpt. temperature
out	*dhdp*	J/mol/Pa - Specific enthalpy differential wrpt. pressure
out	*dhdx*	J/mol - Specific enthalpy differential wrpt. mole numbers
in	*residual*	Set to true if only residual entropy is required

5.16.2.3 entropy()

```
subroutine, public eos::entropy (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(1:nc), intent(in) x,
             integer, intent(in) phase,
             real, intent(out) s,
             real, intent(out), optional dsdt,
             real, intent(out), optional dsdp,
             real, dimension(1:nc), intent(out), optional dsdx,
             logical, intent(in), optional residual )
```

Calculate single-phase specific entropy given composition, temperature and pressure.

Author

MH, 2012-03-20

Parameters

in	*phase*	Phase identifyer
in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*x*	Compozition
out	*s*	J/mol/K - Specific entropy
out	*dsdt*	J/mol/K/K - Specific entropy differential wrpt. temperature
out	*dsdp*	J/mol/K/Pa - Specific entropy differential wrpt. pressure
out	*dsdx*	J/mol/K - Specific entropy differential wrpt. mole numbers
in	*residual*	Set to true if only residual entropy is required

5.16.2.4 getcriticalparam()

```
subroutine, public eos::getcriticalparam (
             integer, intent(in) i,
             real, intent(out) tci,
             real, intent(out) pci,
             real, intent(out) oi,
             real, intent(out), optional vci,
             real, intent(out), optional tnbi )
```

Get critical state (and more) of pure fluid.

Author

MH, 2013-03-06

Parameters

in	*i*	Component index
out	*tci*	K - Critical temperature
out	*pci*	Pa - Critical pressure
out	*oi*	Acentric factor
out	*vci*	m3/mol - Critical volume
out	*tnbi*	Normal boiling point (gs)

5.16.2.5 moleweight()

```
real function, public eos::moleweight (
            real, dimension(1:nc), intent(in) z )
```

Get mole weight. Unit: g/mol.

Author

MH, 2013-03-06

Parameters

in	*z*	Composition

Returns

g/mol - Mole weight

5.16.2.6 pseudo()

```
subroutine, public eos::pseudo (
            real, dimension(1:nc), intent(in) x,
            real, intent(out) tpc,
            real, intent(out) ppc,
            real acfpc,
            real, intent(out) zpc,
            real, intent(out) vpc )
```

Calculate pseudo critical point.

Author

MH, 2012-03-15

Parameters

in	*x*	Compozition
out	*tpc*	K - Pseudo critical temperature
out	*vpc*	m3/mol - Pseudo critical specific volume
out	*ppc*	Pa - Pseudo critical pressure
out	*zpc*	- - Pseudo critical compressibillity

5.16.2.7 pseudo_safe()

```
subroutine, public eos::pseudo_safe (
            real, dimension(1:nc), intent(in) x,
            real, intent(out) tpc,
            real, intent(out) ppc,
            real, intent(out) zpc,
            real, intent(out) vpc )
```

Calculate pseudo critical point, or use estimate from alternative EoS if necessary.

Author

EA, 2015-01

Parameters

in	*x*	Compozition
out	*tpc*	K - Pseudo critical temperature
out	*vpc*	m3/mol - Pseudo critical specific volume
out	*ppc*	Pa - Pseudo critical pressure
out	*zpc*	- - Pseudo critical compressibillity

5.16.2.8 residualgibbs()

```
subroutine, public eos::residualgibbs (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             integer, intent(in) phase,
             real, intent(out) gr,
             real, intent(out), optional dgrdt,
             real, intent(out), optional dgrdp,
             real, dimension(nc), intent(out), optional dgrdn,
             logical, intent(in), optional metaextremum )
```

Calculate residual Gibbs energy. Unit: J/mol.

Author

MH, 2013-10-17

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*z*	Component fractions
in	*phase*	Phase identifyer
out	*gr*	J/mol - Residual Gibbs energy
out	*dgrdt*	J/mol/K - Temperature differential of ideal Gibbs energy
out	*dgrdp*	J/mol/Pa - Pressure differential of ideal Gibbs energy
out	*dgrdn*	J/mol^2 - Mole number differential of ideal Gibbs energy
in	*metaextremum*	Calculate phase properties at metastable extremum

5.16.2.9 specificvolume()

```
subroutine, public eos::specificvolume (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(1:nc), intent(in) x,
             integer, intent(in) phase,
             real, intent(out) v,
             real, intent(out), optional dvdt,
             real, intent(out), optional dvdp,
             real, dimension(1:nc), intent(out), optional dvdx )
```

Calculate single-phase specific volume given composition, temperature and pressure.

Author

MH, 2012-07-06

Parameters

in	*phase*	Phase identifyer
in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*x*	Compozition
out	*v*	m3/mol - Specific volume
out	*dvdt*	m3/mol/K - Specific volume differential wrpt. temperature
out	*dvdp*	m3/mol/Pa - Specific volume differential wrpt. pressure
out	*dvdx*	m3/mol - Specific volume differential wrpt. mole numbers

5.16.2.10 thermo()

```
subroutine, public eos::thermo (
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(1:nc), intent(in) z,
            integer, intent(in) phase,
            real, dimension(1:nc), intent(out) lnfug,
            real, dimension(1:nc), intent(out), optional lnfugt,
            real, dimension(1:nc), intent(out), optional lnfugp,
            real, dimension(1:nc,1:nc), intent(out), optional lnfugx,
            integer, intent(out), optional ophase,
            logical, intent(in), optional metaextremum,
            real, intent(out), optional v )
```

Calculate fugasity coefficient and differentials given composition, temperature and pressure.

Author

MH, 2012-01-27

Parameters

in	*phase*	Phase identifyer
out	*ophase*	Phase identifyer for MINGIBBSPH
in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*z*	Compozition
out	*lnfug*	Logarithm of fugasity coefficient
out	*lnfugt*	1/K - Logarithm of fugasity coefficient differential wrpt. temperature
out	*lnfugp*	1/Pa - Logarithm of fugasity coefficient differential wrpt. pressure
out	*lnfugx*	Logarithm of fugasity coefficient differential wrpt. mole numbers
out	*v*	Specific volume [mol/m3]

5.16.2.11 twophaseenthalpy()

```
real function, public eos::twophaseenthalpy (
            real, intent(in) t,
            real, intent(in) p,
```

```
            real, dimension(1:nc), intent(in) z,
            real, dimension(1:nc), intent(in) x,
            real, dimension(1:nc), intent(in) y,
            real, intent(in) beta,
            integer, intent(in) phase,
            real, intent(in), optional betal )
```

Calculate gas-liquid or single-phase specific enthalpy given composition, temperature and pressure.

Author

MH, 2012-03-20

Parameters

in	*phase*	Phase identifyer
in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*beta*	Gas phase mole fraction
in	*x*	Liquid compozition
in	*y*	Gas compozition
in	*z*	Overall compozition
in	*betal*	Liquid phase mole fraction

Returns

J/mol - Specific mixture enthalpy

5.16.2.12 twophaseentropy()

```
real function, public eos::twophaseentropy (
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(1:nc), intent(in) z,
            real, dimension(1:nc), intent(in) x,
            real, dimension(1:nc), intent(in) y,
            real, intent(in) beta,
            integer, intent(in) phase,
            real, intent(in), optional betal )
```

Calculate gas-liquid or single-phase specific entropy given composition, temperature and pressure.

Author

MH, 2012-03-20

Parameters

in	*phase*	Phase identifyer
in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*beta*	Gas phase mole fraction
in	*x*	Liquid compozition
in	*y*	Gas compozition
in	*z*	Overall compozition
in	*betal*	Liquid phase mole fraction

Returns

J/mol/K - Specific mixture entropy

5.16.2.13 twophaseinternalenergy()

```
real function, public eos::twophaseinternalenergy (
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(1:nc), intent(in) z,
            real, dimension(1:nc), intent(in) x,
            real, dimension(1:nc), intent(in) y,
            real, intent(in) beta,
            integer, intent(in) phase,
            real, intent(in), optional betal )
```

Calculate gas-liquid or single-phase internal energy given composition, temperature, pressure and a phase state given by phase/beta.

Author

EA, 2014-09

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*z*	Overall compozition
in	*x*	Liquid compozition
in	*y*	Gas compozition
in	*beta*	Gas phase mole fraction
in	*phase*	Phase identifier
in	*betal*	Liquid phase mole fraction

Returns

J/mol - Specific internal energy

5.16.2.14 twophasespecificvolume()

```
real function, public eos::twophasespecificvolume (
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(1:nc), intent(in) z,
            real, dimension(1:nc), intent(in) x,
            real, dimension(1:nc), intent(in) y,
            real, intent(in) beta,
            integer, intent(in) phase,
            real, intent(in), optional betal )
```

Calculate gas-liquid or single-phase specific volume given composition, temperature and pressure.

Author

MH, 2012-07-30

Parameters

in	*phase*	Phase identifyer

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*beta*	Gas phase mole fraction
in	*x*	Liquid compozition
in	*y*	Gas compozition
in	*z*	Overall compozition

Returns

m3/mol - Specific mixture volume

Parameters

in	*betal*	Liquid phase mole fraction

5.16.2.15 zfac()

```
subroutine, public eos::zfac (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(1:nc), intent(in) x,
             integer, intent(in) phase,
             real, intent(out) z,
             real, intent(out), optional dzdt,
             real, intent(out), optional dzdp,
             real, dimension(1:nc), intent(out), optional dzdx )
```

Calculate single-phase compressibility factor given composition, temperature and pressure.

Author

MH, 2012-03-20

Parameters

in	*phase*	Phase identifyer
in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*x*	Compozition
out	*z*	Compressibillity factor
out	*dzdt*	1/K - Compressibillity factor differential wrpt. temperature
out	*dzdp*	1/Pa - Compressibillity factor differential wrpt. pressure
out	*dzdx*	Compressibillity factor differential wrpt. mole numbers

5.17 eos_container Module Reference

The module eosdata contains the definitions of the equation of state, mixing rule and the interaction parameters.

Functions/Subroutines

- subroutine allocate_eos (nc, eosstr)

Selection of equation of state and allocation of container classes.

- class(base_eos_param) function, pointer **allocate_p_eos** (nc, eos_index, eos_subindex, eosstr)
- subroutine **assign_thermo_model** (eos_c1, eos_c2)

5.17.1 Detailed Description

The module eosdata contains the definitions of the equation of state, mixing rule and the interaction parameters.

5.17.2 Function/Subroutine Documentation

5.17.2.1 allocate_eos()

```
subroutine eos_container::allocate_eos (
            integer, intent(in) nc,
            character (len=*), intent(in) eosstr )
```

Selection of equation of state and allocation of container classes.

Parameters

eosstr	The equation of state as a character string e.g 'SRK' og 'PR'

The character strings are case-insensitive

Author

Morten Hammer

5.18 eos_parameters Module Reference

The module eosdata contains the definitions of the equation of state, mixing rule and the interaction parameters.

Data Types

- type meos_idealmix
- type single_eos

Functions/Subroutines

- subroutine single_eos_allocate_and_init (eos, nc, eos_label)
- subroutine single_eos_alloc (comp, meos_ptr, eos_label)
- subroutine **single_eos_dealloc** (eos)
- type(single_eos) function **single_eos_constructor** (nc, eos_label)

  Allocate memory for single eos.
- type(meos_idealmix) function **meos_idealmix_constructor** (nc, eos_label)

  Allocate memory for single eos.
- subroutine **assign_single_eos_set** (this, other)
- class(single_eos) function, pointer **get_single_eos_pointer** (eos)

5.18.1 Detailed Description

The module eosdata contains the definitions of the equation of state, mixing rule and the interaction parameters.

5.18.2 Function/Subroutine Documentation

5.18.2.1 single_eos_alloc()

```
subroutine eos_parameters::single_eos_alloc (
            character(len=*), intent(in) comp,
```

```
class(meos), intent(inout), pointer meos_ptr,
character(len=*), intent(in) eos_label )
```

Parameters

in	*eos_label*	EOS label

5.18.2.2 single_eos_allocate_and_init()

```
subroutine eos_parameters::single_eos_allocate_and_init (
             class(single_eos), intent(inout) eos,
             integer, intent(in) nc,
             character(len=*), intent(in) eos_label )
```

Parameters

in	*nc*	Number of components
in	*eos_label*	EOS label

5.19 eosdata Module Reference

The module eosdata contains the definitions of the equation of state, mixing rule and the interaction parameters.

Data Types

- type eos_label_mapping

Functions/Subroutines

- logical function **issrkeos** (subeos_idx)
- logical function **ispreos** (subeos_idx)
- logical function **issafteos** (eos_idx)
- integer function **get_eos_db_idx** (short_label)
- subroutine **get_eos_index** (short_label, eos_index, eos_subindex)
- character(len=short_label_len) function **get_eos_short_label_from_subidx** (subidx)
- integer function **get_eos_idx_from_subidx** (subidx)

Variables

- integer, parameter **eoscubic** = 1

  Cubic model.
- integer, parameter **cbsrk** = 11

  Plain SRK, Soave Redlich Kwong.
- integer, parameter **cbpr** = 12

  Peng-Robinson.
- integer, parameter **cbvdw** = 13

  Van der Waals.
- integer, parameter **cbsw** = 14

  Schmidt-Wensel.
- integer, parameter **cbpt** = 15

  Patel-Teja.
- integer, parameter **eoslk** = 2

  Lee-Kesler.
- integer, parameter **eoscsp** = 3

  Corrensponding State Principle (CSP)
- integer, parameter **cspsrk** = 31

CSP using SRK for scaling.

- integer, parameter **csppr** = 32

  CSP using PR for scaling.

- integer, parameter **eoscpa** = 4

  Cubic Plus Association (CPA)

- integer, parameter **cpasrk** = 41

  SRK Plus Association.

- integer, parameter **cpapr** = 42

  PR Plus Association.

- integer, parameter **eospc_saft** = 5

  PC-SAFT equation of state.

- integer, parameter **eosspc_saft** = 51

  Simplefied PC-SAFT equation of state.

- integer, parameter **eosopc_saft** = 52

  Original PC-SAFT equation of state.

- integer, parameter **eospcp_saft** = 53

  Original PC-SAFT formulation with polar contributions.

- integer, parameter **eosspcp_saft** = 54

  Simplefied PC-SAFT equation of state with polar contributions.

- integer, parameter **eos_single** = 6

  Single component multiparamater eos.

- integer, parameter **meosmbwr19** = 611

  MBWR19 (Bender) multiparameter equation of state.

- integer, parameter **meosmbwr32** = 612

  MBWR32 multiparameter equation of state.

- integer, parameter **meosnist** = 62

  Multiparameter EoS on NIST-like form.

- integer, parameter **meoslj** = 63

  Multiparameter EoS.

- integer, parameter **meosljts** = 64

  Multiparameter EoS.

- integer, parameter **meosgerg** = 65

  Multiparameter EoS.

- integer, parameter **eospt** = 7

  Perturbation theory model.

- integer, parameter **eossaft_vr_mie** = 71

  SAFT-VR-MIE equation of state.

- integer, parameter **eosljs_bh** = 721

  Lennard-Jones splined equation of state using Barker-Henderson perturbation theory.

- integer, parameter **eosljs_wca** = 722

  Lennard-Jones splined equation of state using Weeks-Chandler-Andersen perturbation theory.

- integer, parameter **eosljs_uv** = 723

  Lennard-Jones splined equation of state using Van Westen UV perturbation theory.

- integer, parameter **eosljs_uf** = 724

  Lennard-Jones equation of state using Van Westen UF perturbation theory.

- integer, parameter **eoslj_uf** = 731

  Lennard-Jones equation of state using Van Westen UF perturbation theory.

- integer, parameter **eospets** = 8

  PeTS equation of state for LJTS at 2.5∗sigma.

- integer, parameter **meosnist_mix** = 9

  Multiparameter EoS for fluids with ideal mixture.

- integer, parameter **meosgerg_mix** = 10

  Multicomponent GERG.
- integer, parameter **meos_helm_mix** = 11

  Multicomponent multiparameter EoS with Helmholtz mixing.
- integer, parameter **max_n_eos** = 30
- type(eos_label_mapping), dimension(max_n_eos), parameter **eos_label_db** = (/ eos_label_mapping(eos_idx = eosCubic, eos_subidx = cbSRK, short_label = "SRK", label = "Soave Redlich Kwong", need_alternative_eos = .false.), eos_label_mapping(eos_idx = eosCubic, eos_subidx = cbPR, short_label = "PR", label = "Peng-Robinson", need_alternative_eos = .false.), eos_label_mapping(eos_idx = eosCubic, eos_subidx = cbVdW, short_label = "VDW", label = "van der Waals", need_alternative_eos = .false.), eos_label_mapping(eos_idx = eosCubic, eos_subidx = cbSW, short_label = "SW", label = "Schmidt-Wensel", need_alternative_eos = .false.), eos_label_mapping(eos_idx = eosCubic, eos_subidx = cbPT, short_label = "PT", label = "Patel-Teja", need_alternative_eos = .false.), eos_label_mapping(eos_idx = eosLK, eos_subidx = eosLK, short_label = "LK", label = "Lee-Kesler", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eosCSP, eos_subidx = cspSRK, short_label = "CSP-SRK", label = "Corrensponding state priciple with SRK scaling", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eosCSP, eos_subidx = cspPR, short_label = "CSP-PR", label = "Corrensponding state priciple with PR scaling", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eosCPA, eos_subidx = cpaSRK, short_label = "CPA-SRK", label = "Cubic Pluss Association Soave Redlich Kwong", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eosCPA, eos_subidx = cpaPR, short_label = "CPA-PR", label = "Cubic Pluss Association Peng Robinson", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eosPC_SAFT, eos_subidx = eosSPC_SAFT, short_label = "sPC-SAFT", label = "Simplified Perturbed Chain SAFT", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eosPC_SAFT, eos_subidx = eosPC_SAFT, short_label = "PC-SAFT", label = "Perturbed Chain SAFT", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eosPC_SAFT, eos_subidx = eosSPCP_SAFT, short_label = "sPCP-SAFT", label = "Simplified Perturbed Chain Polar SAFT", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eosPC_SAFT, eos_subidx = eosPCP_SAFT, short_label = "PCP-SAFT", label = "Perturbed Chain Polar SAFT", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eos_single, eos_subidx = meosMbwr19, short_label = "MBWR19", label = "Modified Benedict-Webb-Rubin. Bender 1970.", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eos_single, eos_subidx = meosMbwr32, short_label = "MBWR32", label = "Modified Benedict-Webb-Rubin. Younglove and Ely 1987", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eos_single, eos_subidx = meosNist, short_label = "NIST_MEOS", label = "Multiparameter EoS on NIST-like form", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eos_single, eos_subidx = meosLJ, short_label = "LJ_MEOS", label = "Multiparameter EoS for LJ", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eos_single, eos_subidx = meosLJTS, short_label = "LJTS_MEOS", label = "Multiparameter EoS for LJTS", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eos_single, eos_subidx = meosGERG, short_label = "GERG2008", label = "GERG EoS", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eosPT, eos_subidx = eosSAFT_VR_MIE, short_label = "SAFT-VR-MIE", label = "SAFT for variable range Mie potentials", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eosPT, eos_subidx = eosLJS_BH, short_label = "LJS-BH", label = "LJs equation of state using BH perturbation theory", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eosPT, eos_subidx = eosLJS_WCA, short_label = "LJS-WCA", label = "LJs equation of state using WCA perturbation theory", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eosPT, eos_subidx = eosLJS_UV, short_label = "LJS-UV", label = "LJs equation of state using UV perturbation theory", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eosPT, eos_subidx = eosLJS_UF, short_label = "LJS-UF", label = "LJs equation of state using UF perturbation theory", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eosPT, eos_subidx = eosLJ_UF, short_label = "LJ-UF", label = "LJ equation of state using UF perturbation theory", need_alternative_eos = .true.), eos_label_mapping(eos_idx = eosPeTS, eos_subidx = eosPeTS, short_label = "PETS", label = "PeTS equation of state for LJTS at 2.5*sigma", need_alternative_eos = .true.), eos_label_mapping(eos_idx = meosNist_mix, eos_subidx = meosNist_mix, short_label = "NIST_MEOS_MIX", label = "Ideal mixture of NIST multiparameter EOS", need_alternative_eos = .true.), eos_label_mapping(eos_idx = meosGERG_mix, eos_subidx = meosGERG_mix, short_label = "GERG2008_MIX", label = "GERG2008 mixture model", need_alternative_eos = .true.), eos_label_mapping(eos_idx = meos_helm_mix, eos_subidx = meos_helm_mix, short_label = "MEOS", label = "MEOS mixture model", need_alternative_eos = .true.) /)
- integer, parameter **nsrk** = 3
- integer, dimension(nsrk), parameter **srkindices** = (/ cbSRK, cspSRK, cpaSRK/)
- integer, parameter **npr** = 3

- integer, dimension(npr), parameter **prindices** = (/ cbPR, cspPR, cpaPR/)
- integer, parameter **nsaft** = 4
- integer, dimension(nsaft), parameter **saftindices** = (/ eosCPA, eosPC_SAFT, eosPT, eosPeTS/)

5.19.1 Detailed Description

The module eosdata contains the definitions of the equation of state, mixing rule and the interaction parameters.

5.20 eoslibinit Module Reference

Initialize thermodynamic models.

Functions/Subroutines

- subroutine, public init_volume_translation (volume_trans_model, param_ref)

  Initialize volume translation.
- subroutine, public init_thermo (eos, mixing, alpha, comp_string, nphases, liq_vap_discr_method_in, csp_eos, csp_ref_comp, kij_ref, alpha_ref, saft_ref, b_exponent, trendeosforcp, cptype, silent)

  Initialize thermodynamics library (legacy interface)
- subroutine, public init_cubic (comps, eos, mixing, alpha, parameter_reference, vol_shift)

  Initialize cubic EoS. Use: call init_cubic('CO2,N2','PR', alpha='TWU')
- subroutine, public init_cubic_pseudo (comps, tclist, pclist, acflist, mwlist, mixing, alpha)

  Initialize pseudo components of a cubic EoS. Use: call init_cubic("CO2,PSEUDO,PSEUDO") call init_cubic_↩ pseudo(names=(/"", "C20", "C25"/), Tclist=(\0,300,400), & Pclist=(\0,100e5,200e5), acflist=(\0,0.3,0.5))
- subroutine, public init_tcpr (comps, mixing, parameter_ref)

  Initialize translated and consistent cubic EoS by le Guennec et al.
- subroutine, public init_quantum_cubic (comps, mixing)

  Initialize Quantum Cubic Peng-Robinson equation of state by Aasen et al. (10.1016/j.fluid.2020.112790)
- subroutine, public init_extcsp (comps, sh_eos, sh_mixing, sh_alpha, ref_eos, ref_comp, ref_alpha, parameter_ref)

  Initialize extended corresponding state EoS. Use: call init_extcsp.
- subroutine, public redefine_critical_parameters (silent_init, tc_in, vc_in)

  Set critical parameters to represent actual model.
- subroutine, public init_saftvrmie (comps, parameter_reference)

  Initialize SAFT-VR-MIE EoS. Use: call init_saftvrmie('CO2,N2')
- subroutine, public init_quantum_saftvrmie (comps, feynman_hibbs_order, additive_hs_ref, parameter_↩ reference)

  Initialize SAFT-VR-MIE with quantum corrections EoS. Use: call init_quantum_saftvrmie('He,Ne',feynman_hibbs_↩ order=1)
- subroutine, public init_pcsaft (comps, parameter_reference, simplified, polar)

  Initialize PC-SAFT EoS. Use: call init_pcsaft('CO2,N2')
- subroutine, public init_cpa (comps, eos, mixing, alpha, parameter_reference)

  Initialize CPA EoS. Use: call init_cpa('CO2,N2','PR', alpha='TWU')
- subroutine, public init_lee_kesler (comps, parameter_reference)

  Initialize Lee-Kesler EoS.
- subroutine, public init_multiparameter (comps, meos, ref_state)

  Initialize multiparamaters eos.
- subroutine, public init_pets (parameter_reference)

  Initialize Pets EoS.
- subroutine, public init_ljs (model, parameter_reference)

  Initialize Lennard-Jones splined equation of state using perturbation theory.
- subroutine, public init_lj (model, parameter_reference)

  Initialize Lennard-Jones equation of state using perturbation theory.

Variables

- logical, public **silent_init** = .false.

5.20.1 Detailed Description

Initialize thermodynamic models.

Author

MH, 2014-02

5.20.2 Function/Subroutine Documentation

5.20.2.1 init_cpa()

```
subroutine, public eoslibinit::init_cpa (
             character(len=*), intent(in) comps,
             character(len=*), intent(in), optional eos,
             character(len=*), intent(in), optional mixing,
             character(len=*), intent(in), optional alpha,
             character(len=*), intent(in), optional parameter_reference )
```

Initialize CPA EoS. Use: call init_cpa('CO2,N2','PR', alpha='TWU')

Parameters

in	*comps*	Components. Comma or white-space separated
in	*eos*	Equation of state
in	*mixing*	Mixing rule
in	*alpha*	Alpha correlation
in	*parameter_reference*	Data set reference

5.20.2.2 init_cubic()

```
subroutine, public eoslibinit::init_cubic (
             character(len=*), intent(in) comps,
             character(len=*), intent(in) eos,
             character(len=*), intent(in), optional mixing,
             character(len=*), intent(in), optional alpha,
             character(len=*), intent(in), optional parameter_reference,
             logical, intent(in), optional vol_shift )
```

Initialize cubic EoS. Use: call init_cubic('CO2,N2','PR', alpha='TWU')

Parameters

in	*comps*	Components. Comma or white-space separated
in	*eos*	Equation of state
in	*mixing*	Mixing rule
in	*alpha*	Alpha correlation
in	*parameter_reference*	Parameter reference
in	*vol_shift*	Volume shift

5.20.2.3 init_cubic_pseudo()

```
subroutine, public eoslibinit::init_cubic_pseudo (
             character(len=*), intent(in) comps,
```

```
            real, dimension(nc), intent(in) tclist,
            real, dimension(nc), intent(in) pclist,
            real, dimension(nc), intent(in) acflist,
            real, dimension(nc), intent(in), optional mwlist,
            character(len=*), intent(in), optional mixing,
            character(len=*), intent(in), optional alpha )
```

Initialize pseudo components of a cubic EoS. Use: call init_cubic("CO2,PSEUDO,PSEUDO") call init_cubic_↩ pseudo(names=(/"", "C20", "C25"/), Tclist=(\0,300,400), & Pclist=(\0,100e5,200e5), acflist=(\0,0.3,0.5))

Parameters

in	*comps*	Components. Comma or white-space separated
in	*tclist*	List of critical temperatures (K)
in	*pclist*	List of critical pressures (Pa)
in	*acflist*	List of acentric factors (-)
in	*mwlist*	List of molar masses (kg/mol)
in	*mixing*	Mixing rule
in	*alpha*	Alpha correlation

5.20.2.4 init_extcsp()

```
subroutine, public eoslibinit::init_extcsp (
            character(len=*), intent(in) comps,
            character(len=*), intent(in) sh_eos,
            character(len=*), intent(in) sh_mixing,
            character(len=*), intent(in) sh_alpha,
            character(len=*), intent(in) ref_eos,
            character(len=*), intent(in) ref_comp,
            character(len=*), intent(in), optional ref_alpha,
            character(len=*), intent(in), optional parameter_ref )
```

Initialize extended corresponding state EoS. Use: call init_extcsp.

Parameters

in	*comps*	Components. Comma or white-space
in	*sh_eos*	Shape factor equation of state
in	*sh_alpha*	Shape factor alpha
in	*sh_mixing*	Shape factor mixing rules
in	*ref_eos*	Reference equation of state
in	*ref_comp*	Reference component
in	*ref_alpha*	Needed if refEos is a cubic eos. Should not be present if one want to use an mbwr reference eos.
in	*parameter_ref*	Parameter set reference

5.20.2.5 init_lee_kesler()

```
subroutine, public eoslibinit::init_lee_kesler (
            character(len=*), intent(in) comps,
            character(len=*), intent(in), optional parameter_reference )
```

Initialize Lee-Kesler EoS.

Parameters

in	*comps*	Components. Comma or white-space separated

Parameters

in	*parameter_reference*	Data set reference

5.20.2.6 init_lj()

```
subroutine, public eoslibinit::init_lj (
             character(len=*), intent(in), optional model,
             character(len=*), intent(in), optional parameter_reference )
```

Initialize Lennard-Jones equation of state using perturbation theory.

Parameters

in	*model*	Model selection: "UV" (Default), "UF"
in	*parameter_reference*	Data set reference

5.20.2.7 init_ljs()

```
subroutine, public eoslibinit::init_ljs (
             character(len=*), intent(in), optional model,
             character(len=*), intent(in), optional parameter_reference )
```

Initialize Lennard-Jones splined equation of state using perturbation theory.

Parameters

in	*model*	Model selection: "UV" (Default), "BH", "WCA"
in	*parameter_reference*	Data set reference

5.20.2.8 init_multiparameter()

```
subroutine, public eoslibinit::init_multiparameter (
             character(len=*), intent(in) comps,
             character(len=*), intent(in) meos,
             character(len=*), intent(in) ref_state )
```

Initialize multiparamaters eos.

Parameters

in	*comps*	Components. Comma or white-space separated
in	*meos*	Equation of state
in	*ref_state*	Reference state ("DEFAULT", "IIR", "NBP", "ASHRAE", "IDGAS", "TRIPLE_POINT")

5.20.2.9 init_pcsaft()

```
subroutine, public eoslibinit::init_pcsaft (
             character(len=*), intent(in) comps,
             character(len=*), intent(in), optional parameter_reference,
             logical, intent(in), optional simplified,
             logical, intent(in), optional polar )
```

Initialize PC-SAFT EoS. Use: call init_pcsaft('CO2,N2')

Parameters

in	*comps*	Components. Comma or white-space separated
in	*parameter_reference*	Data set reference
in	*simplified*	Use simplified PC-SAFT (Von Solms et al. 2003: 10.1021/ie020753p)
in	*polar*	Use PCP-SAFT:

5.20.2.10 init_pets()

```
subroutine, public eoslibinit::init_pets (
             character(len=*), intent(in), optional parameter_reference )
```

Initialize Pets EoS.

Parameters

in	*parameter_reference*	Data set reference

5.20.2.11 init_quantum_cubic()

```
subroutine, public eoslibinit::init_quantum_cubic (
             character(len=*), intent(in) comps,
             character(len=*), intent(in), optional mixing )
```

Initialize Quantum Cubic Peng-Robinson equation of state by Aasen et al. (10.1016/j.fluid.2020.112790)

Parameters

in	*comps*	Components. Comma or white-space separated
in	*mixing*	Mixing rule

5.20.2.12 init_quantum_saftvrmie()

```
subroutine, public eoslibinit::init_quantum_saftvrmie (
             character(len=*), intent(in) comps,
             integer, intent(in), optional feynman_hibbs_order,
             logical, intent(in), optional additive_hs_ref,
             character(len=*), intent(in), optional parameter_reference )
```

Initialize SAFT-VR-MIE with quantum corrections EoS. Use: call init_quantum_saftvrmie('He,Ne',feynman_hibbs↩
_order=1)

Parameters

in	*comps*	Components. Comma or white-space separated
in	*parameter_reference*	Data set reference

5.20.2.13 init_saftvrmie()

```
subroutine, public eoslibinit::init_saftvrmie (
             character(len=*), intent(in) comps,
             character(len=*), intent(in), optional parameter_reference )
```

Initialize SAFT-VR-MIE EoS. Use: call init_saftvrmie('CO2,N2')

Parameters

in	*comps*	Components. Comma or white-space separated
in	*parameter_reference*	Data set reference

5.20.2.14 init_tcpr()

```
subroutine, public eoslibinit::init_tcpr (
             character(len=*), intent(in) comps,
             character(len=*), intent(in), optional mixing,
             character(len=*), intent(in), optional parameter_ref )
```

Initialize translated and consistent cubic EoS by le Guennec et al.

Parameters

in	*comps*	Components. Comma or white-space separated
in	*mixing*	Mixing rule
in	*parameter_ref*	Parameter set reference

5.20.2.15 init_thermo()

```
subroutine, public eoslibinit::init_thermo (
             character(len=*), intent(in) eos,
             character(len=*), intent(in) mixing,
             character(len=*), intent(in) alpha,
             character(len=*), intent(in) comp_string,
             integer, intent(in) nphases,
             integer, intent(in), optional liq_vap_discr_method_in,
             character(len=*), intent(in), optional csp_eos,
             character(len=*), intent(in), optional csp_ref_comp,
             character(len=*), intent(in), optional kij_ref,
             character(len=*), intent(in), optional alpha_ref,
             character(len=*), intent(in), optional saft_ref,
             real, intent(in), optional b_exponent,
             character(len=*), intent(in), optional trendeosforcp,
             integer, intent(in), optional cptype,
             logical, intent(in), optional silent )
```

Initialize thermodynamics library (legacy interface)

Author

MH, 2014-02

Parameters

in	*eos*	String defining equation of state
in	*mixing*	String defining mixing rules
in	*alpha*	String defining alpha correlation
in	*comp_string*	String defining components. Comma or white-space separated.
in	*nphases*	Number of phases
in	*liq_vap_discr_method_in*	Method to discriminate between liquid and vapor in case of an undefined single phase. Will be set to none if absent.
in	*csp_eos*	Corrensponding state equation
in	*csp_ref_comp*	CSP component

Parameters

in	*saft_ref*	Data set identifiers
in	*b_exponent*	Inverse exponent (1/s) in mixing of covolume (s>1.0)
in	*trendeosforcp*	Option to init trend for ideal gas properties.
in	*cptype*	Type numbers for Cp
in	*silent*	Option to disable init messages.

5.20.2.16 init_volume_translation()

```
subroutine, public eoslibinit::init_volume_translation (
             character(len=*), intent(in) volume_trans_model,
             character(len=*), intent(in) param_ref )
```

Initialize volume translation.

Author

MH, 2020

Parameters

in	*volume_trans_model*	String model for volume translation
in	*param_ref*	String defining parameter set

5.20.2.17 redefine_critical_parameters()

```
subroutine, public eoslibinit::redefine_critical_parameters (
             logical, intent(in) silent_init,
             real, dimension(nce), intent(in), optional tc_in,
             real, dimension(nce), intent(in), optional vc_in )
```

Set critical parameters to represent actual model.

Author

MH, 2019-03

5.21 eostv Module Reference

Interface to thermodynamic models. Using Helmholtz formulation in temperature and volume.

Functions/Subroutines

- real function, public pressure (t, v, n, dpdv, dpdt, d2pdv2, dpdn, contribution)

  Calculate pressure given composition, temperature and density.
- subroutine, public internal_energy_tv (t, v, n, u, dudt, dudv, dudn, contribution)

  Calculate internal energy given composition, temperature and density.
- subroutine, public free_energy_tv (t, v, n, y, dydt, dydv, dydn, contribution)

  Calculate Helmholtz free energy given composition, temperature and density.
- subroutine, public entropy_tv (t, v, n, s, dsdt, dsdv, dsdn, contribution)

  Calculate entropy given composition, temperature and density.
- subroutine, public enthalpy_tv (t, v, n, h, dhdt, dhdv, dhdn, contribution)

  Calculate enthalpy given composition, temperature and density.
- subroutine, public thermo_tv (t, v, n, lnphi, lnphit, lnphiv, lnphin)

  Calculate fugacity and differentials given composition, temperature and specific volume.

- subroutine, public fres (t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_vv, f_tn, f_vn, f_nn, f_vvv, recalculate)

  Calculate residual reduced Helmholtz energy.
- subroutine, public fres_ne (t, v, ne, f, f_t, f_v, f_n, f_tt, f_tv, f_vv, f_tn, f_vn, f_nn, f_vvv, recalculate)

  Calculate residual reduced Helmholtz energy for real (not apparent) mol numbers.
- subroutine, public fideal (t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_vv, f_tn, f_vn, f_nn)

  Calculate ideal reduced Helmholtz energy.
- subroutine, public fideal_ne (t, v, ne, f, f_t, f_v, f_n, f_tt, f_tv, f_vv, f_tn, f_vn, f_nn)

  Calculate ideal reduced Helmholtz energy for real (not apparent) components.
- subroutine, public virial_coefficients (t, n, b, c)

  Calculate (composition-dependent) virial coefficients B and C, defined as P/RT = rho + B∗rho∗∗2 + C∗rho∗∗3 + O(rho∗∗4) as rho->0.
- subroutine, public secondvirialcoeffmatrix (t, bmat)

  Calculate composition-independent virial coefficients B, defined as P = RT∗rho + B∗rho∗∗2 + C∗rho∗∗3 + O(rho∗∗4) as rho->0. Including cross coefficients.
- subroutine, public binarythirdvirialcoeffmatrix (t, cmat)

  Calculate composition-independent virial coefficients C, defined as P = RT∗rho + B∗rho∗∗2 + C∗rho∗∗3 + O(rho∗∗4) as rho->0. Including cross coefficients Currently the code only support binary mixtures.
- subroutine, public chemical_potential_tv (t, v, n, mu, dmudt, dmudv, dmudn, contribution)

  Calculate chemical potential and derivatives.
- subroutine, public thermo_tvp (t, v, n, lnfug, dlnfugdt, dlnfugdp, dlnfugdn)

  Calculate the logarithmic fugacity coefficient and its differentials. Evaluate using (T,v,n) but output differentials for (T, P, n)
- subroutine, public enthalpy_tvp (t, v, n, h, dhdt, dhdp, dhdn, contribution)

  Calculate enthalpy given composition, temperature and density. Differentials at constant pressure.
- subroutine, public entropy_tvp (t, v, n, s, dsdt, dsdp, dsdn, contribution)

  Calculate entropy given composition, temperature and density. Differentials at constant pressure.

5.21.1 Detailed Description

Interface to thermodynamic models. Using Helmholtz formulation in temperature and volume.

Author

MH, 2015-02

5.21.2 Function/Subroutine Documentation

5.21.2.1 binarythirdvirialcoeffmatrix()

```
subroutine, public eostv::binarythirdvirialcoeffmatrix (
             real, intent(in) t,
             real, dimension(nc,nc), intent(out) cmat )
```

Calculate composition-independent virial coefficients C, defined as P = RT∗rho + B∗rho∗∗2 + C∗rho∗∗3 + O(rho∗∗4) as rho->0. Including cross coefficients Currently the code only support binary mixtures.

Parameters

in	*t*	Temperature [K]
out	*cmat*	Third virial coefficients [m6/mol2]

5.21.2.2 chemical_potential_tv()

```
subroutine, public eostv::chemical_potential_tv (
             real, intent(in) t,
             real, intent(in) v,
```

```
            real, dimension(nc), intent(in) n,
            real, dimension(nc), intent(out) mu,
            real, dimension(nc), intent(out), optional dmudt,
            real, dimension(nc), intent(out), optional dmudv,
            real, dimension(nc,nc), intent(out), optional dmudn,
            integer, intent(in), optional contribution )
```

Calculate chemical potential and derivatives.

Author

MAG, 2018-10-31

Parameters

in	*t*	K - Temperature
in	*v*	m3 - Molar volume
in	*n*	mol - Mol numbers
out	*mu*	J/mol
out	*dmudv*	J/m^3
out	*dmudt*	J/mol K
out	*dmudn*	J/mol^2
in	*contribution*	Contribution from ideal (PROP_IDEAL), residual (PROP_RESIDUAL) or both (PROP_OVERALL)

5.21.2.3 enthalpy_tv()

```
subroutine, public eostv::enthalpy_tv (
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(1:nc), intent(in) n,
            real, intent(out) h,
            real, intent(out), optional dhdt,
            real, intent(out), optional dhdv,
            real, dimension(nc), intent(out), optional dhdn,
            integer, intent(in), optional contribution )
```

Calculate enthalpy given composition, temperature and density.

Author

MH, 2019-06

Parameters

in	*t*	K - Temperature
in	*v*	m3 - Volume
in	*n*	Mol numbers
out	*h*	J - Enthalpy
out	*dhdt*	J/K - Enthalpy differential wrpt. temperature
out	*dhdv*	J/m3 - Enthalpy differential wrpt. volume
out	*dhdn*	J/m3 - Enthalpy differential wrpt. mol numbers
in	*contribution*	Contribution from ideal (PROP_IDEAL), residual (PROP_RESIDUAL) or both (PROP_OVERALL)

5.21.2.4 enthalpy_tvp()

```
subroutine, public eostv::enthalpy_tvp (
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(1:nc), intent(in) n,
             real, intent(out) h,
             real, intent(out), optional dhdt,
             real, intent(out), optional dhdp,
             real, dimension(nc), intent(out), optional dhdn,
             integer, intent(in), optional contribution )
```

Calculate enthalpy given composition, temperature and density. Differentials at constant pressure.

Author

MH, 2019-06

Parameters

in	*t*	K - Temperature
in	*v*	m3 - Volume
in	*n*	Mol numbers
out	*h*	J - Enthalpy
out	*dhdt*	J/K - Enthalpy differential wrpt. temperature (const pressure)
out	*dhdp*	J/Pa - Enthalpy differential wrpt. pressure
out	*dhdn*	J/m3 - Enthalpy differential wrpt. mol numbers
in	*contribution*	Contribution from ideal (PROP_IDEAL), residual (PROP_RESIDUAL) or both (PROP_OVERALL)

5.21.2.5 entropy_tv()

```
subroutine, public eostv::entropy_tv (
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(1:nc), intent(in) n,
             real, intent(out) s,
             real, intent(out), optional dsdt,
             real, intent(out), optional dsdv,
             real, dimension(nc), intent(out), optional dsdn,
             integer, intent(in), optional contribution )
```

Calculate entropy given composition, temperature and density.

Author

MH, 2015-02

Parameters

in	*t*	K - Temperature
in	*v*	m3 - Volume
in	*n*	Mol numbers
out	*s*	J/K - Entropy
out	*dsdt*	J/K2 - Entropy differential wrpt. temperature
out	*dsdv*	J/K/m3 - Entropy differential wrpt. specific volume
out	*dsdn*	J/K/mol - Entropy differential wrpt. mol numbers
in	*contribution*	Contribution from ideal (PROP_IDEAL), residual (PROP_RESIDUAL) or both (PROP_OVERALL)

5.21.2.6 entropy_tvp()

```
subroutine, public eostv::entropy_tvp (
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(1:nc), intent(in) n,
             real, intent(out) s,
             real, intent(out), optional dsdt,
             real, intent(out), optional dsdp,
             real, dimension(nc), intent(out), optional dsdn,
             integer, intent(in), optional contribution )
```

Calculate entropy given composition, temperature and density. Differentials at constant pressure.

Author

MH, 2022-02

Parameters

in	*t*	K - Temperature
in	*v*	m3 - Volume
in	*n*	Mol numbers
out	*s*	J/K - Entropy
out	*dsdt*	J/K2 - Entropy differential wrpt. temperature (const pressure)
out	*dsdp*	J/K/Pa - Entropy differential wrpt. specific pressure
out	*dsdn*	J/K/mol - Entropy differential wrpt. mol numbers
in	*contribution*	Contribution from ideal (PROP_IDEAL), residual (PROP_RESIDUAL) or both (PROP_OVERALL)

5.21.2.7 fideal()

```
subroutine, public eostv::fideal (
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(1:nc), intent(in) n,
             real, intent(out), optional f,
             real, intent(out), optional f_t,
             real, intent(out), optional f_v,
             real, dimension(1:nc), intent(out), optional f_n,
             real, intent(out), optional f_tt,
             real, intent(out), optional f_tv,
             real, intent(out), optional f_vv,
             real, dimension(1:nc), intent(out), optional f_tn,
             real, dimension(1:nc), intent(out), optional f_vn,
             real, dimension(1:nc,1:nc), intent(out), optional f_nn )
```

Calculate ideal reduced Helmholtz energy.

Author

MH, 2012-01-27

Parameters

in	*t*	K - Temperature
in	*v*	m3/mol - Volume
in	*n*	Compozition

5.21.2.8 fideal_ne()

```
subroutine, public eostv::fideal_ne (
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(1:nce), intent(in) ne,
             real, intent(out), optional f,
             real, intent(out), optional f_t,
             real, intent(out), optional f_v,
             real, dimension(1:nce), intent(out), optional f_n,
             real, intent(out), optional f_tt,
             real, intent(out), optional f_tv,
             real, intent(out), optional f_vv,
             real, dimension(1:nce), intent(out), optional f_tn,
             real, dimension(1:nce), intent(out), optional f_vn,
             real, dimension(1:nce,1:nce), intent(out), optional f_nn )
```

Calculate ideal reduced Helmholtz energy for real (not apparent) components.

Author

MH, 2022-02

Parameters

in	*t*	K - Temperature
in	*v*	m3/mol - Volume
in	*ne*	Compozition

5.21.2.9 free_energy_tv()

```
subroutine, public eostv::free_energy_tv (
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(1:nc), intent(in) n,
             real, intent(out) y,
             real, intent(out), optional dydt,
             real, intent(out), optional dydv,
             real, dimension(nc), intent(out), optional dydn,
             integer, intent(in), optional contribution )
```

Calculate Helmholtz free energy given composition, temperature and density.

Author

GL, 2015-01-23

Parameters

in	*t*	K - Temperature
in	*v*	m3 - Volume
in	*n*	Mol numbers
out	*y*	J - Free energy
out	*dydt*	J/K - Differential wrt. temperature
out	*dydv*	J/m3 - Differential wrt. specific volume
out	*dydn*	J/mol - Helmholtz differential wrpt. mol numbers
in	*contribution*	Contribution from ideal (PROP_IDEAL), residual (PROP_RESIDUAL) or both (PROP_OVERALL)

5.21.2.10 fres()

```
subroutine, public eostv::fres (
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(1:nc), intent(in) n,
             real, intent(out), optional f,
             real, intent(out), optional f_t,
             real, intent(out), optional f_v,
             real, dimension(1:nc), intent(out), optional f_n,
             real, intent(out), optional f_tt,
             real, intent(out), optional f_tv,
             real, intent(out), optional f_vv,
             real, dimension(1:nc), intent(out), optional f_tn,
             real, dimension(1:nc), intent(out), optional f_vn,
             real, dimension(1:nc,1:nc), intent(out), optional f_nn,
             real, intent(out), optional f_vvv,
             logical, intent(in), optional recalculate )
```

Calculate residual reduced Helmholtz energy.

Author

MH, 2012-01-27

Parameters

in	*t*	K - Temperature
in	*v*	m3 - Volume
in	*n*	mol numbers

5.21.2.11 fres_ne()

```
subroutine, public eostv::fres_ne (
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(1:nce), intent(in) ne,
             real, intent(out), optional f,
             real, intent(out), optional f_t,
             real, intent(out), optional f_v,
             real, dimension(1:nce), intent(out), optional f_n,
             real, intent(out), optional f_tt,
             real, intent(out), optional f_tv,
             real, intent(out), optional f_vv,
             real, dimension(1:nce), intent(out), optional f_tn,
             real, dimension(1:nce), intent(out), optional f_vn,
             real, dimension(1:nce,1:nce), intent(out), optional f_nn,
             real, intent(out), optional f_vvv,
             logical, intent(in), optional recalculate )
```

Calculate residual reduced Helmholtz energy for real (not apparent) mol numbers.

Author

MH, 2022-02

Parameters

in	*t*	K - Temperature
in	*v*	m3 - Volume
in	*ne*	mol numbers

5.21.2.12 internal_energy_tv()

```
subroutine, public eostv::internal_energy_tv (
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(1:nc), intent(in) n,
             real, intent(out) u,
             real, intent(out), optional dudt,
             real, intent(out), optional dudv,
             real, dimension(nc), intent(out), optional dudn,
             integer, intent(in), optional contribution )
```

Calculate internal energy given composition, temperature and density.

Author

MH, 2012-03-14

Parameters

in	*t*	K - Temperature
in	*v*	m3 - Specific volume
in	*n*	Mol numbers
out	*u*	J - Specific internal energy
out	*dudt*	J/K - Energy differential wrpt. temperature
out	*dudv*	J/m3 - Energy differential wrpt. volume
out	*dudn*	J/mol - Energy differential wrpt. mol numbers
in	*contribution*	Contribution from ideal (PROP_IDEAL), residual (PROP_RESIDUAL) or both (PROP_OVERALL)

5.21.2.13 pressure()

```
real function, public eostv::pressure (
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(1:nc), intent(in) n,
             real, intent(out), optional dpdv,
             real, intent(out), optional dpdt,
             real, intent(out), optional d2pdv2,
             real, dimension(nc), intent(out), optional dpdn,
             integer, intent(in), optional contribution )
```

Calculate pressure given composition, temperature and density.

Author

MH, 2012-03-14

Parameters

in	*t*	K - Temperature
in	*v*	m3 - Volume
in	*n*	Mol numbers
out	*dpdt*	Pa/K - Pressure differential wrpt. temperature
out	*dpdv*	Pa/m3 - Pressure differential wrpt. specific volume
out	*d2pdv2*	Pa/m6 - Second pressure differential wrpt. specific volume
out	*dpdn*	Pa/mol - Second pressure differential wrpt. specific mole numbers
in	*contribution*	Contribution from ideal (PROP_IDEAL), residual (PROP_RESIDUAL) or both (PROP_OVERALL)

Returns

Pa - Pressure

5.21.2.14 secondvirialcoeffmatrix()

```
subroutine, public eostv::secondvirialcoeffmatrix (
            real, intent(in) t,
            real, dimension(nc,nc), intent(out) bmat )
```

Calculate composition-independent virial coefficients B, defined as P = RT∗rho + B∗rho∗∗2 + C∗rho∗∗3 + O(rho∗∗4) as rho->0. Including cross coefficients.

Parameters

in	*t*	Temperature [K]
out	*bmat*	Second virial coefficients [m3/mol]

5.21.2.15 thermo_tv()

```
subroutine, public eostv::thermo_tv (
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(1:nc), intent(in) n,
            real, dimension(1:nc), intent(out) lnphi,
            real, dimension(1:nc), intent(out), optional lnphit,
            real, dimension(1:nc), intent(out), optional lnphiv,
            real, dimension(1:nc,1:nc), intent(out), optional lnphin )
```

Calculate fugacity and differentials given composition, temperature and specific volume.

Author

MH, 2015-10

Parameters

in	*t*	K - Temperature
in	*v*	m3 - Volume
in	*n*	Mole numbers [mol]
out	*lnphi*	Logarithm of fugasity
out	*lnphit*	1/K - Logarithm of fugasity differential wrpt. temperature
out	*lnphiv*	mol/m3 - Logarithm of fugasity differential wrpt. volume
out	*lnphin*	Logarithm of fugasity differential wrpt. mole numbers

5.21.2.16 thermo_tvp()

```
subroutine, public eostv::thermo_tvp (
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(nc), intent(in) n,
            real, dimension(nc), intent(out) lnfug,
            real, dimension(nc), intent(out), optional dlnfugdt,
            real, dimension(nc), intent(out), optional dlnfugdp,
            real, dimension(nc,nc), intent(out), optional dlnfugdn )
```

Calculate the logarithmic fugacity coefficient and its differentials. Evaluate using (T,v,n) but output differentials for (T, P, n)

Author

Morten Hammer, 2022-02

Parameters

in	*t*	Temperature [K]
in	*v*	Volume [m3]
in	*n*	Mole numbers [mols]

5.21.2.17 virial_coefficients()

```
subroutine, public eostv::virial_coefficients (
             real, intent(in) t,
             real, dimension(1:nc), intent(in) n,
             real, intent(out) b,
             real, intent(out) c )
```

Calculate (composition-dependent) virial coefficients B and C, defined as P/RT = rho + B∗rho∗∗2 + C∗rho∗∗3 + O(rho∗∗4) as rho->0.

Parameters

in	*t*	Temperature [K]
in	*n*	Composition [-]
out	*b*	Second virial coefficients [m3/mol]
out	*c*	Third virial coefficient [m6/mol2]

5.22 excess_gibbs Module Reference

Excess Gibbs Energy Models.

Functions/Subroutines

- real function, public **getinfinitlimitc** (cbeos)
- real function **getzerolimitc** (cbeos)
- subroutine **getgeparam** (cbeos, t, tau, dtaudt, d2taudt2, cij, alpha)
- subroutine **gethvnrtlparam** (cbeos, t, tau, dtaudt, d2taudt2, cij, alpha)
- subroutine, public **getpoly** (p, n, x, y, yd, ydd)
- subroutine, public getfraction (frac, x, y, yd, ydd)
- subroutine **getwsparam** (cbeos, t, tau, dtaudt, d2taudt2, cij, alpha)
- subroutine **gexcess** (cbeos, t, n, gexinf, dgexinfdt, d2gexinfdt2, dgexinfdni, d2gexinfdnidt, d2gexinfdnidnj)
- subroutine, public **geinf** (cbeos, t, zcomp, gexinf, dgexinfdt, d2gexinfdt2, dgexinfdni, d2gexinfdnidt, d2gexinfdnidnj)
- subroutine, public **excessgibbsmix** (cbeos, t, n)

5.22.1 Detailed Description

Excess Gibbs Energy Models.
Currently support Huron-Vidal and NRTL
Based on Geir Skaugens implementation of Huron-Vidal, and Anders Austegards implementation of Wong-Sandler

Author

Morten Hammer

Date

2015-03

5.22.2 Function/Subroutine Documentation

5.22.2.1 getfraction()

```
subroutine, public excess_gibbs::getfraction (
             type(fraction), intent(in) frac,
             real, intent(in) x,
             real, intent(out) y,
             real, intent(out) yd,
             real, intent(out) ydd )
```

Parameters

out	*ydd*	y and its derived

5.23 extcsp Module Reference

Calulate shape factors for corresponding state method, using either a cubic or a multiparameter EoS for the reference EoS, and a cubic EoS for the shape factor EoS.

Data Types

- type extcsp_eos
- type shape_diff

  Derivatives. Uses the notation on pages 115-120 in Michelsen & Mollerup.

Functions/Subroutines

- subroutine, public csp_init (eos, nce, comps, refcomp_str, sheos, shmixrule, shalpha, refeos, refalpha, parameter_ref)

  Init calculation for shape factors, by setting index of reference component.
- subroutine, public csp_zfac (eos, t, p, n, phase, zfac, dzdt, dzdp, dzdz)

  Calculate Z-factor of the mixture.
- subroutine, public csp_refpressure (eos, t0, v0, n, p0, dp0dv0, dp0dt0)

  Solve reference equation for p0 for given real temperature and pressure.
- subroutine, public **csp_maintestroutine** ()
- subroutine, public csp_testpressure (t, v, n)
- subroutine, public **csp_calcfres** (nce, eos, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_vv, f_tn, f_vn, f_nn)

  Calculates the reduced residual Helmholtz energy F, along with its derivatives.

5.23.1 Detailed Description

Calulate shape factors for corresponding state method, using either a cubic or a multiparameter EoS for the reference EoS, and a cubic EoS for the shape factor EoS.

Todo Need trace-component functionality.

This module uses the convention that all quantities are measured in base SI units, with the exception of density [mol/L] and molar volume [L/mol].

5.23.2 Function/Subroutine Documentation

5.23.2.1 csp_init()

```
subroutine, public extcsp::csp_init (
             class(extcsp_eos), intent(inout) eos,
             integer, intent(in) nce,
             type(gendata_pointer), dimension(:), intent(inout), allocatable comps,
             character(len=*), intent(in) refcomp_str,
             character(len=*), intent(in) sheos,
             character(len=*), intent(in) shmixrule,
             character(len=*), intent(in) shalpha,
             character(len=*), intent(in) refeos,
             character(len=*), intent(in), optional refalpha,
             character(len=*), intent(in), optional parameter_ref )
```

Init calculation for shape factors, by setting index of reference component.

Author

MHA, 2013-11-27

Ailo

Parameters

in	*refcomp_str*	Reference component
in	*refeos*	shEos is any two-parameter (a,b) cubic equation of state
in	*refalpha*	Needed if refEos is a cubic eos. Should not be present if one want to use an mbwr reference eos.
in	*parameter_ref*	Parameter set reference

5.23.2.2 csp_refpressure()

```
subroutine, public extcsp::csp_refpressure (
             class(extcsp_eos), intent(inout) eos,
             real, intent(in) t0,
             real, intent(in) v0,
             real, dimension(nce), intent(in) n,
             real, intent(out) p0,
             real, intent(out), optional dp0dv0,
             real, intent(out), optional dp0dt0 )
```

Solve reference equation for p0 for given real temperature and pressure.

Author

MH, 2013-11-27

Parameters

in,out	*eos*	CSP eos
in	*t0*	Reference fluid temperature [K]
in	*v0*	Reference fluid molar volume [L/mol]
in	*n*	Mole composition of mixture [mol]
out	*p0*	Pressure [Pa]
out	*dp0dt0*	Pressure differentials [Pa∗mol/L], [Pa/K]

5.23.2.3 csp_testpressure()

```
subroutine, public extcsp::csp_testpressure (
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(nce), intent(in) n )
```

Parameters

in	*t*	Mixture temperature [K]
in	*v*	Mixture molar volume [L/mol]
in	*n*	Mole composition of mixture [mol]

5.23.2.4 csp_zfac()

```
subroutine, public extcsp::csp_zfac (
             class(extcsp_eos), intent(inout) eos,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nce), intent(in) n,
             integer, intent(in) phase,
             real, intent(out) zfac,
             real, intent(out), optional dzdt,
             real, intent(out), optional dzdp,
             real, dimension(nce), intent(out), optional dzdz )
```

Calculate Z-factor of the mixture.

Author

MH, 2013-11-27

Parameters

in,out	*eos*	CSP eos
in	*p*	Volume shape factor [mol]
in	*t*	Temperature [K]
in	*n*	Molar values [mol]
out	*zfac*	[-]

5.24 gergdatadb Module Reference

Automatically generated file gergdatadb.f90 Time stamp: 2021-07-25T15:05:48.938109.

Data Types

- type gergdata

Variables

- type(gergdata), parameter **gerg_1** = gergdata(ident = "AR", name = "ARGON", mw = 39.948, tc = 150.↩ 687, pc = 4879.8, rhoc = 13.407429659, ttr = 83.8058, ptr = 69.03, tr = 150.687, rhor = 13.407429659, Rgas = 8.314472, acf = -0.0006, t_max = 700.0, p_max = 1000000.0, n_eos = 12, a_eos = (/ 0.↩ 85095714803969,-0.24003222943480d1,0.54127841476466, 0.16919770692538d-1,0.68825965019035d-1,0.21428032815338d-3, 0.17429895321992,-0.33654495604194d-1,-0.13526799857691, -0.16387350791552d-1,-0.24987666851475d-1,0.88769204815709d-2, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,

0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.250,1.125,1.500, 1.375,0.250,0.875, 0.625,1.750,3.625, 3.625,14.5,12.↩
0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7,
2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0 /), n_cosh =
0, n_sinh = 0, n_id = (/ 8.3166315,-4.9465026,1.5, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.,1.,1.,0.0d0,0.↩
0d0,0.0d0,0.0d0 /))

- type(gergdata), parameter **gerg_2** = gergdata(ident = "CO", name = "CO", mw = 28.0101, tc = 132.86, pc = 3494.0, rhoc = 10.85, ttr = 68.16, ptr = 15.45, tr = 132.86, rhor = 10.85, Rgas = 8.314472, acf = 0.↩
0497, t_max = 500.0, p_max = 100000.0, n_eos = 12, a_eos = (/ 0.90554,-2.4515,0.53149, 0.024173,0.↩
072156,0.00018818, 0.19405,-0.043268,-0.12778, -0.027896,-0.034154,0.016329, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.125,1.5, 1.375,0.25,0.875, 0.↩
625,1.75,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3,
0,0,0,0,0,0, 0,0,0,0,0,0 /), n_cosh = 1, n_sinh = 1, n_id = (/ 10.813340744,-19.834733959,2.50055, -0.↩
00493,1.02865,0.0d0, 0.0d0 /), t_id = (/ 0.,1.,1.,5.302762306,11.6698028,0.0d0,0.0d0 /))
- type(gergdata), parameter **gerg_3** = gergdata(ident = "H2O", name = "WATER", mw = 18.01528, tc = 647.096, pc = 22064.0, rhoc = 17.87371609, ttr = 273.16, ptr = 0.61248, tr = 647.096, rhor = 17.↩
87371609, Rgas = 8.314472, acf = 0.345, t_max = 1350.0, p_max = 1000000.0, n_eos = 16, a↩
_eos = (/ 0.82728408749586,-0.18602220416584d1,-0.11199009613744d1, 0.15635753976056,0.↩
87375844859025,-0.36674403715731, 0.53987893432436d-1,0.10957690214499d1,0.53213037828563d-
1, 0.13050533930825d-1,-0.41079520434476,0.14637443344120, -0.55726838623719d-1,-0.11201774143800d-
1,-0.66062758068099d-2, 0.46918522004538d-2,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /),
t_eos = (/ 0.5,1.25,1.875, 0.125,1.5,1.0, 0.75,1.5,0.625, 2.625,5.0,4.0, 4.5,3.0,4.0, 6.0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,2,3, 4,1,5,5,1,2, 4,4,1,1,0,0, 0,0,0,0,0,0 /), l_eos
= (/ 0,0,0,0,0,0, 0,1,1,1,2,2, 2,3,5,5,0,0, 0,0,0,0,0,0 /), n_cosh = 1, n_sinh = 2, n_id = (/ 8.20352069,-11.↩
996306443,3.00392, -0.98763,0.01059,3.06904, 0.0d0 /), t_id = (/ 0.,1.,1.,1.763895929,0.415386589,3.↩
874803739,0.0d0 /))
- type(gergdata), parameter **gerg_4** = gergdata(ident = "CO2", name = "CO2", mw = 44.0095, tc = 304.1282, pc = 7377.3, rhoc = 10.624978698, ttr = 216.592, ptr = 517.94, tr = 304.1282, rhor = 10.624978698, Rgas = 8.314472, acf = 0.225, t_max = 1100.0, p_max = 800000.0, n_eos = 22, a_eos = (/ 0.52646564804653,-0.14995725042592d1,0.27329786733782, 0.12949500022786,0.↩
15404088341841,-0.58186950946814, -0.18022494838296,-0.95389904072812d-1,-0.80486819317679d-
2, -0.35547751273090d-1,-0.28079014882405,-0.82435890081677d-1, 0.10832427979006d-1,-0.↩
67073993161097d-2,-0.46827907600524d-2, -0.28359911832177d-1,0.19500174744098d-1,-0.21609137507166,
0.43772794926972,-0.22130790113593,0.15190189957331d-1, -0.15380948953300d-1,0.0d0,0.0d0 /), t↩
_eos = (/ 0.00,1.25,1.625, 0.375,0.375,1.375, 1.125,1.375,0.125, 1.625,3.75,3.5, 7.5,8.0,6.0, 16.0,11.0,24.0,
26.0,28.0,24.0, 26.0,0.0d0,0.0d0 /), d_eos = (/ 1,1,2,3,3,3, 4,5,6,6,1,4, 1,1,3,3,4,5, 5,5,5,5,0,0 /), l_eos = (/
0,0,0,0,1,1, 1,1,1,1,2,2, 3,3,3,3,3,5, 5,5,6,6,0,0 /), n_cosh = 2, n_sinh = 2, n_id = (/ 11.925152758,-16.↩
118762264,2.50002, 1.06044,-0.01393,2.04452, 2.03366 /), t_id = (/ 0.,1.,1.,-2.844425476,1.12159609,3.↩
022758166,1.589964364 /))
- type(gergdata), parameter **gerg_5** = gergdata(ident = "NC10", name = "DECANE", mw = 142.28168, tc = 617.7, pc = 2103.0, rhoc = 1.64, ttr = 243.5, ptr = 0.0014, tr = 617.7, rhor = 1.64, Rgas = 8.314472, acf = 0.4884, t_max = 675.0, p_max = 800000.0, n_eos = 12, a_eos = (/ 1.0461,-2.4807,0.74372, -0.↩
52579,0.15315,0.00032865, 0.84178,0.055424,-0.73555, -0.18507,-0.020775,0.012335, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.125,1.5, 1.375,0.25,0.875, 0.↩
625,1.75,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3,
0,0,0,0,0,0, 0,0,0,0,0,0 /), n_cosh = 1, n_sinh = 2, n_id = (/ 15.870791919,-108.858547525,3., -43.4931,21.↩
0069,58.3657, 0.0d0 /), t_id = (/ 0.,1.,1.,1.353835195,0.267034159,2.833479035,0.0d0 /))
- type(gergdata), parameter **gerg_6** = gergdata(ident = "O2", name = "OXYGEN", mw = 31.9988, tc = 154.595, pc = 5061.6, rhoc = 13.63, ttr = 54.361, ptr = 0.1460, tr = 154.595, rhor = 13.63, Rgas = 8.314472, acf = 0.0236, t_max = 1000.0, p_max = 82000.0, n_eos = 12, a_eos = (/ 0.↩
88878286369701,-0.24879433312148d1,0.59750190775886, 0.96501817061881d-2,0.71970428712770d-
1,0.22337443000195d-3, 0.18558686391474,-0.38129368035760d-1,-0.15352245383006, -0.26726814910919d-
1,-0.25675298677127d-1,0.95714302123668d-2, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.250,1.125,1.500, 1.375,0.250,0.875, 0.625,1.750,3.625, 3.625,14.5,12.↩
0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7,
2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0 /), n_cosh =
1, n_sinh = 1, n_id = (/ 10.001843586,-14.996095135,2.50146, -1.01334,1.07558,0.0d0, 0.0d0 /), t_id = (/

0.,1.,1.,7.223325463,14.461722565,0.0d0,0.0d0 /))

- type(gergdata), parameter **gerg_7** = gergdata(ident = "NC8", name = "OCTANE", mw = 114.22852, tc = 569.32, pc = 2506.7, rhoc = 2.056404127, ttr = 216.37, ptr = 0.001989, tr = 569.32, rhor = 2.056404127, Rgas = 8.314472, acf = 0.3964, t_max = 600.0, p_max = 100000.0, n_eos = 12, a_eos = (/ 0.10722544875633d1,-0.24632951172003d1,0.65386674054928, -0.36324974085628,0.12713269626764,0.30713572777930d-3, 0.52656856987540,0.19362862857653d-1,-0.58939426849155, -0.14069963991934,-0.78966330500036d-2,0.33036597968109d-2, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.250,1.125,1.500, 1.375,0.250,0.875, 0.625,1.750,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0 /), n_cosh = 1, n_sinh = 2, n_id = (/ 15.864687161,-97.370667555,3., -33.8029,15.6865,48.1731, 0.0d0 /), t_id = (/ 0.,1.,1.,1.431644769,0.27914354,2.973845992,0.0d0 /))
- type(gergdata), parameter **gerg_8** = gergdata(ident = "IC4", name = "ISOBUTAN", mw = 58.1222, tc = 407.817, pc = 3633.1, rhoc = 3.86014294, ttr = 113.73, ptr = 0.00002177, tr = 407.817, rhor = 3.86014294, Rgas = 8.314472, acf = 0.1841, t_max = 575.0, p_max = 35000.0, n_eos = 12, a_eos = (/ 0.10429331589100d1,-0.28184272548892d1,0.86176232397850, -0.10613619452487,0.98615749302134d-1,0.23948208682322d-3, 0.30330004856950,-0.41598156135099d-1,-0.29991937470058, -0.80369342764109d-1,-0.29761373251151d-1,0.13059630303140d-1, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.250,1.125,1.500, 1.375,0.250,0.875, 0.625,1.750,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0 /), n_cosh = 2, n_sinh = 2, n_id = (/ 20.413726078,-94.467620036,3.06714, -5.25156,-16.1388,8.97575, 25.1423 /), t_id = (/ 0.,1.,1.,0.485556021,2.19158348,1.074673199,4.671261865 /))
- type(gergdata), parameter **gerg_9** = gergdata(ident = "HE", name = "HELIUM", mw = 4.002602, tc = 5.1953, pc = 227.46, rhoc = 17.399, ttr = 2.1768, ptr = 4.8565, tr = 5.1953, rhor = 17.399, Rgas = 8.314472, acf = -0.3859, t_max = 1500.0, p_max = 100000.0, n_eos = 12, a_eos = (/ -0.45579024006737,0.12516390754925d1,-0.15438231650621d1, 0.20467489707221d-1,-0.34476212380781,-0.20858459512787d-1, 0.16227414711778d-1,-0.57471818200892d-1,0.19462416430715d-1, -0.33295680123020d-1,-0.10863577372367d-1,-0.22173365245954d-1, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.0,0.125,0.75, 1.0,0.75,2.625, 0.125,1.25,2.0, 1.0,4.5,5.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,4,1,3, 5,5,5,2,1,2, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,1,1, 1,1,1,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0 /), n_cosh = 0, n_sinh = 0, n_id = (/ 13.628409737,-143.470759602,1.5, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.,1.,1.,0.0d0,0.0d0,0.0d0,0.0d0 /))
- type(gergdata), parameter **gerg_10** = gergdata(ident = "H2", name = "HYDROGEN", mw = 2.01588, tc = 33.19, pc = 1315.0, rhoc = 14.94, ttr = 13.957, ptr = 6.669, tr = 33.19, rhor = 14.94, Rgas = 8.314472, acf = -0.2187, t_max = 400.0, p_max = 121000.0, n_eos = 14, a_eos = (/ 0.53579928451252d1,-0.62050252530595d1,0.13830241327086, -0.71397954896129d-1,0.15474053959733d-1,-0.14976806405771, -0.26368723988451d-1,0.56681303156066d-1,-0.60063958030436d-1, -0.45043942027132,0.42478840244500,-0.21997640827139d-1, -0.10499521374530d-1,-0.28955902866816d-2,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.50,0.625,0.375, 0.625,1.125,2.625, 0.0,0.25,1.375, 4.0,4.25,5.0, 8.0,8.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,2,2,4,1, 5,5,5,1,1,2, 5,1,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,1, 1,1,1,2,2,3, 3,5,0,0,0,0, 0,0,0,0,0,0 /), n_cosh = 2, n_sinh = 2, n_id = (/ 13.796443393,-175.864487294,1.47906, -0.45444,1.3756,0.95806, 1.56039 /), t_id = (/ 0.,1.,1.,9.84763483,50.367279301,6.891654113,49.76529075 /))
- type(gergdata), parameter **gerg_11** = gergdata(ident = "NC5", name = "PENTANE", mw = 72.14878, tc = 469.7, pc = 3368.8, rhoc = 3.215577588, ttr = 143.47, ptr = 0.00007632, tr = 469.7, rhor = 3.215577588, Rgas = 8.314472, acf = 0.2513, t_max = 600.0, p_max = 100000.0, n_eos = 12, a_eos = (/ 0.10968643098001d1,-0.29988888298061d1,0.99516886799212, -0.16170708558539,0.11334460072775,0.26760595150748d-3, 0.40979881986931,-0.40876423083075d-1,-0.38169482469447, -0.10931956843993,-0.32073223327990d-1,0.16877016216975d-1, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.250,1.125,1.500, 1.375,0.250,0.875, 0.625,1.750,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0 /), n_cosh = 1, n_sinh = 2, n_id = (/ 14.536611217,-89.919548319,3., -21.836,8.95043,33.4032, 0.0d0 /), t_id = (/ 0.,1.,1.,1.789520971,0.380391739,3.777411113,0.0d0 /))

- type(gergdata), parameter **gerg_12** = gergdata(ident = "NC4", name = "BUTANE", mw = 58.1222, tc = 425.125, pc = 3830.3, rhoc = 3.920016792, ttr = 134.895, ptr = 0.0006507, tr = 425.125, rhor = 3.920016792, Rgas = 8.314472, acf = 0.2038, t_max = 575.0, p_max = 69000.0, n_eos = 12, a_eos = (/ 0.10626277411455d1,-0.28620951828350d1,0.88738233403777, -0.12570581155345,0.10286308708106,0.25358040602654d-3, 0.32325200233982,-0.37950761057432d-1,-0.32534802014452, -0.79050969051011d-1,-0.20636720547775d-1,0.57053809334750d-2, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.250,1.125,1.500, 1.375,0.250,0.875, 0.625,1.750,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0 /), n_cosh = 2, n_sinh = 2, n_id = (/ 20.884143364,-91.638478026,3.33944, -6.89406,-14.7824,9.44893, 24.4618 /), t_id = (/ 0.,1.,1.,0.43195766,2.124516319,1.101487798,4.502440459 /))
- type(gergdata), parameter **gerg_13** = gergdata(ident = "N2", name = "NITROGEN", mw = 28.0134, tc = 126.192, pc = 3395.8, rhoc = 11.1839, ttr = 63.151, ptr = 12.523, tr = 126.192, rhor = 11.1839, Rgas = 8.314472, acf = 0.0373, t_max = 2000.0, p_max = 2200000.0, n_eos = 24, a_eos = (/ 0.59889711801201,-0.16941557480731d1,0.24579736191718, -0.23722456755175,0.17954918715141d-1,0.14592875720215d-1, 0.10008065936206,0.73157115385532,-0.88372272336366, 0.31887660246708,0.20766491728799,-0.19379315454158d-1, -0.16936641554983,0.13546846041701,-0.33066712095307d-1, -0.60690817018557d-1,0.12797548292871d-1,0.58743664107299d-2, -0.18451951971969d-1,0.47226622042472d-2,-0.52024079680599d-2, 0.43563505956635d-1,-0.36251690750939d-1,-0.28974026866543d-2 /), t_eos = (/ 0.125,1.125,0.375, 1.125,0.625,1.5, 0.625,2.625,2.75, 2.125,2.0,1.75, 4.50,4.75,5.0, 4.0,4.5,7.5, 14.0,11.5,26.0, 28.0,30.0,16.0 /), d_eos = (/ 1,1,2,2,4,4, 1,1,1,2,3,6, 2,3,3,4,4,2, 3,4,5,6,6,7 /), l_eos = (/ 0,0,0,0,0,0, 1,1,1,1,1,1, 2,2,2,2,2,3, 3,3,6,6,6,6 /), n_cosh = 1, n_sinh = 2, n_id = (/ 11.083407489,-22.202102428,2.50031, 0.1466,0.13732,0.90066, 0.0d0 /), t_id = (/ 0.,1.,1.,-5.393067706,5.25182262,13.788988208,0.0d0 /))
- type(gergdata), parameter **gerg_14** = gergdata(ident = "NC7", name = "HEPTANE", mw = 100.20194, tc = 540.13, pc = 2773.8, rhoc = 2.315324434, ttr = 182.55, ptr = 0.0001755, tr = 540.13, rhor = 2.315324434, Rgas = 8.314472, acf = 0.3554, t_max = 600.0, p_max = 100000.0, n_eos = 12, a_eos = (/ 0.10543747645262d1,-0.26500681506144d1,0.81730047827543, -0.30451391253428,0.12253868710800,0.27266472743928d-3, 0.49865825681670,-0.71432815084176d-3,-0.54236895525450, -0.13801821610756,-0.61595287380011d-2,0.48602510393022d-3, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.250,1.125,1.500, 1.375,0.250,0.875, 0.625,1.750,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0 /), n_cosh = 1, n_sinh = 2, n_id = (/ 15.063786601,-97.345252349,3., -30.4707,13.7266,43.5561, 0.0d0 /), t_id = (/ 0.,1.,1.,1.54813656,0.314348398,3.259326458,0.0d0 /))
- type(gergdata), parameter **gerg_15** = gergdata(ident = "C2", name = "ETHANE", mw = 30.06904, tc = 305.322, pc = 4871.8, rhoc = 6.87085454, ttr = 90.368, ptr = 0.001113, tr = 305.322, rhor = 6.87085454, Rgas = 8.314472, acf = 0.0995, t_max = 675.0, p_max = 900000.0, n_eos = 24, a_eos = (/ 0.63596780450714,-0.17377981785459d1,0.28914060926272, -0.33714276845694,0.22405964699561d-1,0.15715424886913d-1, 0.11450634253745,0.10612049379745d1,-0.12855224439423d1, 0.39414630777652,0.31390924682041,-0.21592277117247d-1, -0.21723666564905,-0.28999574439489,0.42321173025732, 0.46434100259260d-1,-0.13138398329741,0.11492850364368d-1, -0.33387688429909d-1,0.15183171583644d-1,-0.47610805647657d-2, 0.46917166277885d-1,-0.39401755804649d-1,-0.32569956247611d-2 /), t_eos = (/ 0.125,1.125,0.375, 1.125,0.625,1.5, 0.625,2.625,2.75, 2.125,2.0,1.75, 4.5,4.75,5.0, 4.0,4.5,7.5, 14.0,11.5,26.0, 28.0,30.0,16.0 /), d_eos = (/ 1,1,2,2,4,4, 1,1,1,2,3,6, 2,3,3,4,4,2, 3,4,5,6,6,7 /), l_eos = (/ 0,0,0,0,0,0, 1,1,1,1,1,1, 2,2,2,2,2,3, 3,3,6,6,6,6 /), n_cosh = 2, n_sinh = 2, n_id = (/ 24.675437527,-77.42531376,3.00263, -1.23722,6.01989,4.33939, 13.1974 /), t_id = (/ 0.,1.,1.,0.731306621,3.508721939,1.831882406,3.378007481 /))
- type(gergdata), parameter **gerg_16** = gergdata(ident = "H2S", name = "H2S", mw = 34.08088, tc = 373.1, pc = 9000.0, rhoc = 10.19, ttr = 187.7, ptr = 23.3, tr = 373.1, rhor = 10.19, Rgas = 8.314472, acf = 0.1005, t_max = 760.0, p_max = 170000.0, n_eos = 12, a_eos = (/ 0.87641,-2.0367,0.21634, -0.050199,0.066994,0.00019076, 0.20227,-0.0045348,-0.22230, -0.034714,-0.014885,0.0074154, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.125,1.5, 1.375,0.25,0.875, 0.625,1.75,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), n_cosh = 1, n_sinh = 1, n_id = (/ 9.336197742,-16.266508995,3., -1.00243,3.11942,0.0d0, 0.0d0 /), t_id = (/ 0.,1.,1.,2.27065398,4.914580541,0.0d0,0.0d0 /))
- type(gergdata), parameter **gerg_17** = gergdata(ident = "IC5", name = "IPENTANE", mw = 72.14878, tc =

460.35, pc = 3378.0, rhoc = 3.271, ttr = 112.65, ptr = 0.83D-7, tr = 460.35, rhor = 3.271, Rgas = 8.↩
314472, acf = 0.2274, t_max = 500.0, p_max = 1000000.0, n_eos = 12, a_eos = (/ 1.0963,-3.0402,1.0317, -
0.15410,0.11535,0.00029809, 0.39571,-0.045881,-0.35804, -0.10107,-0.035484,0.018156, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.125,1.5, 1.375,0.25,0.875,
0.625,1.75,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3,
0,0,0,0,0,0, 0,0,0,0,0,0 /), n_cosh = 1, n_sinh = 2, n_id = (/ 15.449907693,-101.298172792,3., -20.1101,11.↩
7618,33.1688, 0.0d0 /), t_id = (/ 0.,1.,1.,1.977271641,0.635392636,4.169371131,0.0d0 /))

- type(gergdata), parameter **gerg_18** = gergdata(ident = "C1", name = "METHANE", mw = 16.04246, tc = 190.564, pc = 4599.2, rhoc = 10.139342719, ttr = 90.6941, ptr = 11.698, tr = 190.564, rhor = 10.↩
139342719, Rgas = 8.314472, acf = 0.0114, t_max = 625.0, p_max = 1000000.0, n_eos = 24, a_eos = (/
0.57335704239162,-0.16760687523730d1,0.23405291834916, -0.21947376343441,0.16369201404128d-
1,0.15004406389280d-1, 0.98990489492918d-1,0.58382770929055,-0.74786867560390, 0.30033302857974,0.↩
20985543806568,-0.18590151133061d-1, -0.15782558339049,0.12716735220791,-0.32019743894346d-
1, -0.68049729364536d-1,0.24291412853736d-1,0.51440451639444d-2, -0.19084949733532d-1,0.↩
55229677241291d-2,-0.44197392976085d-2, 0.40061416708429d-1,-0.33752085907575d-1,-0.25127658213357d-
2 /), t_eos = (/ 0.125,1.125,0.375, 1.125,0.625,1.5, 0.625,2.625,2.75, 2.125,2.0,1.75, 4.50,4.75,5.00, 4.↩
00,4.50,7.50, 14.0,11.5,26.0, 28.0,30.0,16.0 /), d_eos = (/ 1,1,2,2,4,4, 1,1,1,2,3,6, 2,3,3,4,4,2, 3,4,5,6,6,7
/), l_eos = (/ 0,0,0,0,0,0, 1,1,1,1,1,1, 2,2,2,2,2,3, 3,3,6,6,6,6 /), n_cosh = 2, n_sinh = 2, n_id = (/ 19.↩
597508817,-83.959667892,3.00088, -0.0046,4.46921,0.76315, 8.74432 /), t_id = (/ 0.,1.,1.,0.936220902,5.↩
722644361,4.306474465,5.577233895 /))
- type(gergdata), parameter **gerg_19** = gergdata(ident = "NC6", name = "HEXANE", mw = 86.17536, tc = 507.82, pc = 3041.7, rhoc = 2.705877875, ttr = 177.83, ptr = 0.001277, tr = 507.82, rhor = 2.↩
705877875, Rgas = 8.314472, acf = 0.3000, t_max = 600.0, p_max = 100000.0, n_eos = 12, a↩
_eos = (/ 0.10553238013661d1,-0.26120615890629d1,0.76613882967260, -0.29770320622459,0.↩
11879907733358,0.27922861062617d-3, 0.46347589844105,0.11433196980297d-1,-0.48256968738131,
-0.93750558924659d-1,-0.67273247155994d-2,-0.51141583585428d-2, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.250,1.125,1.500, 1.375,0.250,0.875,
0.625,1.750,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3,
0,0,0,0,0,0, 0,0,0,0,0,0 /), n_cosh = 1, n_sinh = 2, n_id = (/ 14.345969349,-96.165722367,3., -26.8142,11.↩
6977,38.6164, 0.0d0 /), t_id = (/ 0.,1.,1.,1.691951873,0.359036667,3.596924107,0.0d0 /))
- type(gergdata), parameter **gerg_20** = gergdata(ident = "C3", name = "PROPANE", mw = 44.09562, tc = 369.825, pc = 4255.5, rhoc = 5.000043088, ttr = 85.48, ptr = 0.00000017, tr = 369.825, rhor = 5.↩
000043088, Rgas = 8.314472, acf = 0.1538, t_max = 500.0, p_max = 100000.0, n_eos = 12, a_↩
eos = (/ 0.10403973107358d1,-0.28318404081403d1,0.84393809606294, -0.76559591850023d-1,0.↩
94697373057280d-1,0.24796475497006d-3, 0.27743760422870,-0.43846000648377d-1,-0.26991064784350,
-0.69313413089860d-1,-0.29632145981653d-1,0.14040126751380d-1, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.250,1.125,1.500, 1.375,0.250,0.875,
0.625,1.750,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3,
0,0,0,0,0,0, 0,0,0,0,0,0 /), n_cosh = 2, n_sinh = 2, n_id = (/ 31.602908195,-84.463284382,3.02939, -3.↩
197,8.37267,6.60569, 19.1921 /), t_id = (/ 0.,1.,1.,0.543210978,2.777773271,1.297521801,2.583146083 /)
)
- type(gergdata), parameter **gerg_21** = gergdata(ident = "NC9", name = "NONANE", mw = 128.2551, tc = 594.55, pc = 2281.0, rhoc = 1.81, ttr = 219.7, ptr = 0.00044, tr = 594.55, rhor = 1.81, Rgas = 8.314472, acf = 0.4433, t_max = 600.0, p_max = 800000.0, n_eos = 12, a_eos = (/ 1.1151,-2.7020,0.83416, -0.38828,0.↩
13760,0.00028185, 0.62037,0.015847,-0.61726, -0.15043,-0.012982,0.0044325, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.125,1.5, 1.375,0.25,0.875, 0.↩
625,1.75,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3,
0,0,0,0,0,0, 0,0,0,0,0,0 /), n_cosh = 1, n_sinh = 2, n_id = (/ 16.313913248,-102.160247463,3., -38.1235,18.↩
0241,53.3415, 0.0d0 /), t_id = (/ 0.,1.,1.,1.370586158,0.263819696,2.848860483,0.0d0 /))
- integer, parameter **maxgerg** = 21
- type(gergdata), dimension(maxgerg), parameter **gergdb** = (/ gerg_1,gerg_2,gerg_3,gerg_4,gerg_5,gerg↩
_6, gerg_7,gerg_8,gerg_9,gerg_10,gerg_11,gerg_12, gerg_13,gerg_14,gerg_15,gerg_16,gerg_17,gerg_18,
gerg_19,gerg_20,gerg_21 /)

5.24.1 Detailed Description

Automatically generated file gergdatadb.f90 Time stamp: 2021-07-25T15:05:48.938109.

5.25 gergmixdb Module Reference

Automatically generated file gergmixdb.f90 Time stamp: 2022-10-03T21:42:11.071422.

Data Types

- type gerg_mix_data
- type gerg_mix_reducing

Variables

- type(gerg_mix_reducing), parameter **gerg_red_1** = gerg_mix_reducing(ident1 = "CO2", ident2 = "H2", beta_v = 0.904142159, gamma_v = 1.15279255, beta_T = 0.942320195, gamma_T = 1.782924792)
- type(gerg_mix_reducing), parameter **gerg_red_2** = gerg_mix_reducing(ident1 = "NC10", ident2 = "O2", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_3** = gerg_mix_reducing(ident1 = "C1", ident2 = "H2S", beta_v = 1.012599087, gamma_v = 1.040161207, beta_T = 1.011090031, gamma_T = 0.961155729)
- type(gerg_mix_reducing), parameter **gerg_red_4** = gerg_mix_reducing(ident1 = "N2", ident2 = "NC6", beta_v = 1., gamma_v = 1.195952177, beta_T = 1., gamma_T = 1.472607971)
- type(gerg_mix_reducing), parameter **gerg_red_5** = gerg_mix_reducing(ident1 = "C2", ident2 = "C3", beta_v = 0.997607277, gamma_v = 1.00303472, beta_T = 0.996199694, gamma_T = 1.01473019)
- type(gerg_mix_reducing), parameter **gerg_red_6** = gerg_mix_reducing(ident1 = "CO2", ident2 = "NC9", beta_v = 1., gamma_v = 0.973386152, beta_T = 1.00768862, gamma_T = 1.140671202)
- type(gerg_mix_reducing), parameter **gerg_red_7** = gerg_mix_reducing(ident1 = "NC7", ident2 = "NC8", beta_v = 1., gamma_v = 1.006767176, beta_T = 1., gamma_T = 0.998793111)
- type(gerg_mix_reducing), parameter **gerg_red_8** = gerg_mix_reducing(ident1 = "NC6", ident2 = "NC10", beta_v = 1.001516371, gamma_v = 1.013511439, beta_T = 0.99764101, gamma_T = 1.028939539)
- type(gerg_mix_reducing), parameter **gerg_red_9** = gerg_mix_reducing(ident1 = "NC4", ident2 = "NC7", beta_v = 1., gamma_v = 1.019174227, beta_T = 1., gamma_T = 1.021283378)
- type(gerg_mix_reducing), parameter **gerg_red_10** = gerg_mix_reducing(ident1 = "CO2", ident2 = "NC10", beta_v = 1.000151132, gamma_v = 1.183394668, beta_T = 1.02002879, gamma_T = 1.145512213)
- type(gerg_mix_reducing), parameter **gerg_red_11** = gerg_mix_reducing(ident1 = "C2", ident2 = "IC4", beta_v = 1., gamma_v = 1.006616886, beta_T = 1., gamma_T = 1.033283811)
- type(gerg_mix_reducing), parameter **gerg_red_12** = gerg_mix_reducing(ident1 = "C2", ident2 = "NC6", beta_v = 1., gamma_v = 1.169701102, beta_T = 1., gamma_T = 1.092177796)
- type(gerg_mix_reducing), parameter **gerg_red_13** = gerg_mix_reducing(ident1 = "IC4", ident2 = "IC5", beta_v = 1., gamma_v = 1.002284353, beta_T = 1., gamma_T = 1.001835788)
- type(gerg_mix_reducing), parameter **gerg_red_14** = gerg_mix_reducing(ident1 = "C3", ident2 = "NC10", beta_v = 0.984104227, gamma_v = 1.053040574, beta_T = 0.985331233, gamma_T = 1.140905252)
- type(gerg_mix_reducing), parameter **gerg_red_15** = gerg_mix_reducing(ident1 = "H2", ident2 = "H2S", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_16** = gerg_mix_reducing(ident1 = "C1", ident2 = "O2", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 0.95)
- type(gerg_mix_reducing), parameter **gerg_red_17** = gerg_mix_reducing(ident1 = "H2", ident2 = "CO", beta_v = 1., gamma_v = 1.121416201, beta_T = 1., gamma_T = 1.377504607)
- type(gerg_mix_reducing), parameter **gerg_red_18** = gerg_mix_reducing(ident1 = "C1", ident2 = "NC4", beta_v = 0.979105972, gamma_v = 1.045375122, beta_T = 0.99417491, gamma_T = 1.171607691)
- type(gerg_mix_reducing), parameter **gerg_red_19** = gerg_mix_reducing(ident1 = "CO", ident2 = "HE", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_20** = gerg_mix_reducing(ident1 = "IC4", ident2 = "NC10", beta_v = 1., gamma_v = 1.060243344, beta_T = 1., gamma_T = 1.021624748)
- type(gerg_mix_reducing), parameter **gerg_red_21** = gerg_mix_reducing(ident1 = "C1", ident2 = "N2", beta_v = 0.998721377, gamma_v = 1.013950311, beta_T = 0.99809883, gamma_T = 0.979273013)

- type(gerg_mix_reducing), parameter **gerg_red_22** = gerg_mix_reducing(ident1 = "CO2", ident2 = "NC6", beta_v = 1., gamma_v = 0.851343711, beta_T = 1., gamma_T = 1.038675574)
- type(gerg_mix_reducing), parameter **gerg_red_23** = gerg_mix_reducing(ident1 = "NC4", ident2 = "IC4", beta_v = 1.000880464, gamma_v = 1.00041444, beta_T = 1.000077547, gamma_T = 1.001432824)
- type(gerg_mix_reducing), parameter **gerg_red_24** = gerg_mix_reducing(ident1 = "NC6", ident2 = "NC8", beta_v = 1., gamma_v = 1.006268954, beta_T = 1., gamma_T = 1.001633952)
- type(gerg_mix_reducing), parameter **gerg_red_25** = gerg_mix_reducing(ident1 = "C2", ident2 = "NC7", beta_v = 1., gamma_v = 1.057666085, beta_T = 1., gamma_T = 1.134532014)
- type(gerg_mix_reducing), parameter **gerg_red_26** = gerg_mix_reducing(ident1 = "C3", ident2 = "AR", beta↩_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_27** = gerg_mix_reducing(ident1 = "NC10", ident2 = "H2S", beta_v = 0.975187766, gamma_v = 1.171714677, beta_T = 0.973091413, gamma_T = 1.103693489)
- type(gerg_mix_reducing), parameter **gerg_red_28** = gerg_mix_reducing(ident1 = "NC5", ident2 = "NC6", beta_v = 1., gamma_v = 1.002480637, beta_T = 1., gamma_T = 1.000761237)
- type(gerg_mix_reducing), parameter **gerg_red_29** = gerg_mix_reducing(ident1 = "C3", ident2 = "H2O", beta_v = 1., gamma_v = 1.011759763, beta_T = 1., gamma_T = 0.600340961)
- type(gerg_mix_reducing), parameter **gerg_red_30** = gerg_mix_reducing(ident1 = "NC6", ident2 = "H2O", beta_v = 1., gamma_v = 1.170217596, beta_T = 1., gamma_T = 0.569681333)
- type(gerg_mix_reducing), parameter **gerg_red_31** = gerg_mix_reducing(ident1 = "NC4", ident2 = "IC5", beta_v = 1., gamma_v = 1.002728434, beta_T = 1., gamma_T = 1.000792201)
- type(gerg_mix_reducing), parameter **gerg_red_32** = gerg_mix_reducing(ident1 = "NC4", ident2 = "H2O", beta_v = 1., gamma_v = 1.223638763, beta_T = 1., gamma_T = 0.615512682)
- type(gerg_mix_reducing), parameter **gerg_red_33** = gerg_mix_reducing(ident1 = "N2", ident2 = "C2", beta↩_v = 0.978880168, gamma_v = 1.042352891, beta_T = 1.007671428, gamma_T = 1.098650964)
- type(gerg_mix_reducing), parameter **gerg_red_34** = gerg_mix_reducing(ident1 = "IC5", ident2 = "H2S", beta_v = 1., gamma_v = 0.835763343, beta_T = 1., gamma_T = 0.982651529)
- type(gerg_mix_reducing), parameter **gerg_red_35** = gerg_mix_reducing(ident1 = "NC10", ident2 = "H2O", beta_v = 1., gamma_v = 0.551405318, beta_T = 0.897162268, gamma_T = 0.740416402)
- type(gerg_mix_reducing), parameter **gerg_red_36** = gerg_mix_reducing(ident1 = "NC6", ident2 = "NC7", beta_v = 1., gamma_v = 1.001508227, beta_T = 1., gamma_T = 0.999762786)
- type(gerg_mix_reducing), parameter **gerg_red_37** = gerg_mix_reducing(ident1 = "IC4", ident2 = "NC7", beta_v = 1., gamma_v = 1.021668316, beta_T = 1., gamma_T = 1.00988576)
- type(gerg_mix_reducing), parameter **gerg_red_38** = gerg_mix_reducing(ident1 = "NC9", ident2 = "O2", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_39** = gerg_mix_reducing(ident1 = "C1", ident2 = "NC8", beta_v = 0.994740603, gamma_v = 1.116549372, beta_T = 0.957473785, gamma_T = 1.449245409)
- type(gerg_mix_reducing), parameter **gerg_red_40** = gerg_mix_reducing(ident1 = "NC9", ident2 = "HE", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_41** = gerg_mix_reducing(ident1 = "NC9", ident2 = "H2", beta_v = 1., gamma_v = 1.342647661, beta_T = 1., gamma_T = 2.23435404)
- type(gerg_mix_reducing), parameter **gerg_red_42** = gerg_mix_reducing(ident1 = "NC4", ident2 = "CO", beta_v = 1., gamma_v = 1.084740904, beta_T = 1., gamma_T = 1.173916162)
- type(gerg_mix_reducing), parameter **gerg_red_43** = gerg_mix_reducing(ident1 = "NC9", ident2 = "H2O", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_44** = gerg_mix_reducing(ident1 = "NC4", ident2 = "NC8", beta_v = 1., gamma_v = 1.046905515, beta_T = 1., gamma_T = 1.033180106)
- type(gerg_mix_reducing), parameter **gerg_red_45** = gerg_mix_reducing(ident1 = "C2", ident2 = "NC4", beta_v = 0.999157205, gamma_v = 1.006179146, beta_T = 0.999130554, gamma_T = 1.034832749)
- type(gerg_mix_reducing), parameter **gerg_red_46** = gerg_mix_reducing(ident1 = "NC10", ident2 = "H2", beta_v = 1.695358382, gamma_v = 1.120233729, beta_T = 1.064818089, gamma_T = 3.786003724)
- type(gerg_mix_reducing), parameter **gerg_red_47** = gerg_mix_reducing(ident1 = "NC7", ident2 = "H2S", beta_v = 0.828967164, gamma_v = 1.087956749, beta_T = 0.988937417, gamma_T = 1.013453092)
- type(gerg_mix_reducing), parameter **gerg_red_48** = gerg_mix_reducing(ident1 = "IC4", ident2 = "H2S", beta_v = 1.012994431, gamma_v = 0.988591117, beta_T = 0.974550548, gamma_T = 0.937130844)
- type(gerg_mix_reducing), parameter **gerg_red_49** = gerg_mix_reducing(ident1 = "NC7", ident2 = "AR", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)

- type(gerg_mix_reducing), parameter **gerg_red_50** = gerg_mix_reducing(ident1 = "N2", ident2 = "NC5", beta_v = 1., gamma_v = 1.078877166, beta_T = 1., gamma_T = 1.419029041)
- type(gerg_mix_reducing), parameter **gerg_red_51** = gerg_mix_reducing(ident1 = "NC5", ident2 = "NC7", beta_v = 1., gamma_v = 1.008972412, beta_T = 1., gamma_T = 1.002441051)
- type(gerg_mix_reducing), parameter **gerg_red_52** = gerg_mix_reducing(ident1 = "CO2", ident2 = "H2S", beta_v = 0.906630564, gamma_v = 1.024085837, beta_T = 1.016034583, gamma_T = 0.92601888)
- type(gerg_mix_reducing), parameter **gerg_red_53** = gerg_mix_reducing(ident1 = "NC6", ident2 = "H2", beta_v = 1., gamma_v = 1.243461678, beta_T = 1., gamma_T = 3.021197546)
- type(gerg_mix_reducing), parameter **gerg_red_54** = gerg_mix_reducing(ident1 = "C2", ident2 = "HE", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_55** = gerg_mix_reducing(ident1 = "NC8", ident2 = "NC9", beta_v = 1., gamma_v = 1.001357085, beta_T = 1., gamma_T = 1.000235044)
- type(gerg_mix_reducing), parameter **gerg_red_56** = gerg_mix_reducing(ident1 = "HE", ident2 = "AR", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_57** = gerg_mix_reducing(ident1 = "CO", ident2 = "H2O", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_58** = gerg_mix_reducing(ident1 = "NC9", ident2 = "CO", beta_v = 1., gamma_v = 1.252151449, beta_T = 1., gamma_T = 1.294070556)
- type(gerg_mix_reducing), parameter **gerg_red_59** = gerg_mix_reducing(ident1 = "O2", ident2 = "HE", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_60** = gerg_mix_reducing(ident1 = "CO2", ident2 = "AR", beta_v = 1.008392428, gamma_v = 1.029205465, beta_T = 0.996512863, gamma_T = 1.050971635)
- type(gerg_mix_reducing), parameter **gerg_red_61** = gerg_mix_reducing(ident1 = "N2", ident2 = "NC10", beta_v = 1., gamma_v = 1., beta_T = 0.957934447, gamma_T = 1.822157123)
- type(gerg_mix_reducing), parameter **gerg_red_62** = gerg_mix_reducing(ident1 = "N2", ident2 = "CO2", beta_v = 0.977794634, gamma_v = 1.047578256, beta_T = 1.005894529, gamma_T = 1.107654104)
- type(gerg_mix_reducing), parameter **gerg_red_63** = gerg_mix_reducing(ident1 = "C2", ident2 = "NC5", beta_v = 0.993851009, gamma_v = 1.026085655, beta_T = 0.998688946, gamma_T = 1.066665676)
- type(gerg_mix_reducing), parameter **gerg_red_64** = gerg_mix_reducing(ident1 = "C1", ident2 = "NC7", beta_v = 0.962050831, gamma_v = 1.156655935, beta_T = 0.977431529, gamma_T = 1.379850328)
- type(gerg_mix_reducing), parameter **gerg_red_65** = gerg_mix_reducing(ident1 = "C1", ident2 = "AR", beta_v = 1.034630259, gamma_v = 1.014678542, beta_T = 0.990954281, gamma_T = 0.989843388)
- type(gerg_mix_reducing), parameter **gerg_red_66** = gerg_mix_reducing(ident1 = "H2O", ident2 = "AR", beta_v = 1., gamma_v = 1.038993495, beta_T = 1., gamma_T = 1.070941866)
- type(gerg_mix_reducing), parameter **gerg_red_67** = gerg_mix_reducing(ident1 = "C3", ident2 = "H2", beta_v = 1., gamma_v = 1.07400611, beta_T = 1., gamma_T = 2.308215191)
- type(gerg_mix_reducing), parameter **gerg_red_68** = gerg_mix_reducing(ident1 = "NC8", ident2 = "CO", beta_v = 1., gamma_v = 1.219206702, beta_T = 1., gamma_T = 1.276565536)
- type(gerg_mix_reducing), parameter **gerg_red_69** = gerg_mix_reducing(ident1 = "NC10", ident2 = "HE", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_70** = gerg_mix_reducing(ident1 = "CO2", ident2 = "NC5", beta_v = 1.024311498, gamma_v = 1.068406078, beta_T = 1.027000795, gamma_T = 0.979217302)
- type(gerg_mix_reducing), parameter **gerg_red_71** = gerg_mix_reducing(ident1 = "NC5", ident2 = "H2", beta_v = 1., gamma_v = 1.188334783, beta_T = 1., gamma_T = 2.013859174)
- type(gerg_mix_reducing), parameter **gerg_red_72** = gerg_mix_reducing(ident1 = "IC5", ident2 = "NC9", beta_v = 1., gamma_v = 1.028994325, beta_T = 1., gamma_T = 1.008191499)
- type(gerg_mix_reducing), parameter **gerg_red_73** = gerg_mix_reducing(ident1 = "NC7", ident2 = "NC10", beta_v = 1., gamma_v = 1.002972346, beta_T = 1., gamma_T = 1.002229938)
- type(gerg_mix_reducing), parameter **gerg_red_74** = gerg_mix_reducing(ident1 = "NC5", ident2 = "O2", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_75** = gerg_mix_reducing(ident1 = "C3", ident2 = "H2S", beta_v = 0.936811219, gamma_v = 1.010593999, beta_T = 0.992573556, gamma_T = 0.905829247)
- type(gerg_mix_reducing), parameter **gerg_red_76** = gerg_mix_reducing(ident1 = "NC7", ident2 = "CO", beta_v = 1., gamma_v = 1.190354273, beta_T = 1., gamma_T = 1.256123503)
- type(gerg_mix_reducing), parameter **gerg_red_77** = gerg_mix_reducing(ident1 = "NC4", ident2 = "NC5", beta_v = 1., gamma_v = 1.01815965, beta_T = 1., gamma_T = 1.00214364)

- type(gerg_mix_reducing), parameter **gerg_red_78** = gerg_mix_reducing(ident1 = "N2", ident2 = "O2", beta↩ _v = 0.99952177, gamma_v = 0.997082328, beta_T = 0.997190589, gamma_T = 0.995157044)
- type(gerg_mix_reducing), parameter **gerg_red_79** = gerg_mix_reducing(ident1 = "C3", ident2 = "NC9", beta_v = 1., gamma_v = 1.199769134, beta_T = 1., gamma_T = 1.109973833)
- type(gerg_mix_reducing), parameter **gerg_red_80** = gerg_mix_reducing(ident1 = "N2", ident2 = "IC4", beta↩ _v = 0.98641583, gamma_v = 1.100576129, beta_T = 0.99286813, gamma_T = 1.284462634)
- type(gerg_mix_reducing), parameter **gerg_red_81** = gerg_mix_reducing(ident1 = "H2O", ident2 = "HE", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_82** = gerg_mix_reducing(ident1 = "NC4", ident2 = "H2", beta_v = 1., gamma_v = 1.232939523, beta_T = 1., gamma_T = 2.509259945)
- type(gerg_mix_reducing), parameter **gerg_red_83** = gerg_mix_reducing(ident1 = "N2", ident2 = "H2S", beta_v = 0.910394249, gamma_v = 1.256844157, beta_T = 1.004692366, gamma_T = 0.9601742)
- type(gerg_mix_reducing), parameter **gerg_red_84** = gerg_mix_reducing(ident1 = "C2", ident2 = "IC5", beta↩ _v = 1., gamma_v = 1.045439935, beta_T = 1., gamma_T = 1.021150247)
- type(gerg_mix_reducing), parameter **gerg_red_85** = gerg_mix_reducing(ident1 = "C1", ident2 = "H2O", beta_v = 1.012783169, gamma_v = 1.585018334, beta_T = 1.063333913, gamma_T = 0.775810513)
- type(gerg_mix_reducing), parameter **gerg_red_86** = gerg_mix_reducing(ident1 = "NC5", ident2 = "NC10", beta_v = 1., gamma_v = 1.016370338, beta_T = 1., gamma_T = 1.049035838)
- type(gerg_mix_reducing), parameter **gerg_red_87** = gerg_mix_reducing(ident1 = "O2", ident2 = "AR", beta↩ _v = 0.999746847, gamma_v = 0.993907223, beta_T = 1.000023103, gamma_T = 0.990430423)
- type(gerg_mix_reducing), parameter **gerg_red_88** = gerg_mix_reducing(ident1 = "C3", ident2 = "NC5", beta_v = 1.044919431, gamma_v = 1.019921513, beta_T = 0.996484021, gamma_T = 1.008344412)
- type(gerg_mix_reducing), parameter **gerg_red_89** = gerg_mix_reducing(ident1 = "N2", ident2 = "NC7", beta_v = 1., gamma_v = 1.40455409, beta_T = 1., gamma_T = 1.520975334)
- type(gerg_mix_reducing), parameter **gerg_red_90** = gerg_mix_reducing(ident1 = "IC5", ident2 = "H2", beta↩ _v = 1., gamma_v = 1.184340443, beta_T = 1., gamma_T = 1.996386669)
- type(gerg_mix_reducing), parameter **gerg_red_91** = gerg_mix_reducing(ident1 = "N2", ident2 = "NC8", beta_v = 1., gamma_v = 1.186067025, beta_T = 1., gamma_T = 1.733280051)
- type(gerg_mix_reducing), parameter **gerg_red_92** = gerg_mix_reducing(ident1 = "C1", ident2 = "IC5", beta↩ _v = 1., gamma_v = 1.343685343, beta_T = 1., gamma_T = 1.188899743)
- type(gerg_mix_reducing), parameter **gerg_red_93** = gerg_mix_reducing(ident1 = "NC10", ident2 = "CO", beta_v = 1., gamma_v = 0.87018496, beta_T = 1.049594632, gamma_T = 1.803567587)
- type(gerg_mix_reducing), parameter **gerg_red_94** = gerg_mix_reducing(ident1 = "NC5", ident2 = "IC5", beta_v = 1., gamma_v = 1.000024335, beta_T = 1., gamma_T = 1.000050537)
- type(gerg_mix_reducing), parameter **gerg_red_95** = gerg_mix_reducing(ident1 = "C1", ident2 = "C3", beta↩ _v = 1.00482707, gamma_v = 1.038470657, beta_T = 0.989680305, gamma_T = 1.098655531)
- type(gerg_mix_reducing), parameter **gerg_red_96** = gerg_mix_reducing(ident1 = "NC7", ident2 = "H2", beta_v = 1., gamma_v = 1.159131722, beta_T = 1., gamma_T = 3.169143057)
- type(gerg_mix_reducing), parameter **gerg_red_97** = gerg_mix_reducing(ident1 = "NC6", ident2 = "O2", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_98** = gerg_mix_reducing(ident1 = "NC6", ident2 = "AR", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_99** = gerg_mix_reducing(ident1 = "CO2", ident2 = "NC8", beta_v = 1.026169373, gamma_v = 1.104043935, beta_T = 1.02969078, gamma_T = 1.074455386)
- type(gerg_mix_reducing), parameter **gerg_red_100** = gerg_mix_reducing(ident1 = "CO2", ident2 = "NC7", beta_v = 1.205469976, gamma_v = 1.164585914, beta_T = 1.011806317, gamma_T = 1.046169823)
- type(gerg_mix_reducing), parameter **gerg_red_101** = gerg_mix_reducing(ident1 = "N2", ident2 = "AR", beta_v = 1.004166412, gamma_v = 1.002212182, beta_T = 0.999069843, gamma_T = 0.990034831)
- type(gerg_mix_reducing), parameter **gerg_red_102** = gerg_mix_reducing(ident1 = "C2", ident2 = "H2", beta_v = 0.925367171, gamma_v = 1.10607204, beta_T = 0.932969831, gamma_T = 1.902008495)
- type(gerg_mix_reducing), parameter **gerg_red_103** = gerg_mix_reducing(ident1 = "C1", ident2 = "NC5", beta_v = 0.94833012, gamma_v = 1.124508039, beta_T = 0.992127525, gamma_T = 1.249173968)
- type(gerg_mix_reducing), parameter **gerg_red_104** = gerg_mix_reducing(ident1 = "IC4", ident2 = "NC8", beta_v = 1., gamma_v = 1.032807063, beta_T = 1., gamma_T = 1.013945424)
- type(gerg_mix_reducing), parameter **gerg_red_105** = gerg_mix_reducing(ident1 = "IC4", ident2 = "NC9", beta_v = 1., gamma_v = 1.047298475, beta_T = 1., gamma_T = 1.017817492)

- type(gerg_mix_reducing), parameter **gerg_red_106** = gerg_mix_reducing(ident1 = "H2O", ident2 = "H2S", beta_v = 1., gamma_v = 1.014832832, beta_T = 1., gamma_T = 0.940587083)
- type(gerg_mix_reducing), parameter **gerg_red_107** = gerg_mix_reducing(ident1 = "N2", ident2 = "CO", beta_v = 1., gamma_v = 1.008690943, beta_T = 1., gamma_T = 0.993425388)
- type(gerg_mix_reducing), parameter **gerg_red_108** = gerg_mix_reducing(ident1 = "C2", ident2 = "AR", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_109** = gerg_mix_reducing(ident1 = "IC5", ident2 = "HE", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_110** = gerg_mix_reducing(ident1 = "IC5", ident2 = "H2O", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_111** = gerg_mix_reducing(ident1 = "NC4", ident2 = "AR", beta_v = 1., gamma_v = 1.214638734, beta_T = 1., gamma_T = 1.245039498)
- type(gerg_mix_reducing), parameter **gerg_red_112** = gerg_mix_reducing(ident1 = "C1", ident2 = "NC10", beta_v = 1.033086292, gamma_v = 1.146089637, beta_T = 0.937777823, gamma_T = 1.568231489)
- type(gerg_mix_reducing), parameter **gerg_red_113** = gerg_mix_reducing(ident1 = "IC5", ident2 = "AR", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_114** = gerg_mix_reducing(ident1 = "C1", ident2 = "HE", beta_v = 1., gamma_v = 0.881405683, beta_T = 1., gamma_T = 3.159776855)
- type(gerg_mix_reducing), parameter **gerg_red_115** = gerg_mix_reducing(ident1 = "C3", ident2 = "HE", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_116** = gerg_mix_reducing(ident1 = "IC4", ident2 = "H2", beta_v = 1., gamma_v = 1.147595688, beta_T = 1., gamma_T = 1.895305393)
- type(gerg_mix_reducing), parameter **gerg_red_117** = gerg_mix_reducing(ident1 = "IC5", ident2 = "NC10", beta_v = 1., gamma_v = 1.039372957, beta_T = 1., gamma_T = 1.010825138)
- type(gerg_mix_reducing), parameter **gerg_red_118** = gerg_mix_reducing(ident1 = "O2", ident2 = "H2O", beta_v = 1., gamma_v = 1.143174289, beta_T = 1., gamma_T = 0.964767932)
- type(gerg_mix_reducing), parameter **gerg_red_119** = gerg_mix_reducing(ident1 = "C2", ident2 = "NC9", beta_v = 1., gamma_v = 1.14353473, beta_T = 1., gamma_T = 1.05603303)
- type(gerg_mix_reducing), parameter **gerg_red_120** = gerg_mix_reducing(ident1 = "NC4", ident2 = "NC10", beta_v = 0.976951968, gamma_v = 1.027845529, beta_T = 0.993688386, gamma_T = 1.076466918)
- type(gerg_mix_reducing), parameter **gerg_red_121** = gerg_mix_reducing(ident1 = "NC5", ident2 = "H2O", beta_v = 1., gamma_v = 0.95667731, beta_T = 1., gamma_T = 0.447666011)
- type(gerg_mix_reducing), parameter **gerg_red_122** = gerg_mix_reducing(ident1 = "O2", ident2 = "CO", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_123** = gerg_mix_reducing(ident1 = "H2", ident2 = "AR", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_124** = gerg_mix_reducing(ident1 = "NC4", ident2 = "HE", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_125** = gerg_mix_reducing(ident1 = "C2", ident2 = "O2", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_126** = gerg_mix_reducing(ident1 = "NC9", ident2 = "AR", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_127** = gerg_mix_reducing(ident1 = "NC5", ident2 = "CO", beta_v = 1., gamma_v = 1.119954454, beta_T = 1., gamma_T = 1.206043295)
- type(gerg_mix_reducing), parameter **gerg_red_128** = gerg_mix_reducing(ident1 = "C3", ident2 = "NC8", beta_v = 1., gamma_v = 1.102764612, beta_T = 1., gamma_T = 1.063694129)
- type(gerg_mix_reducing), parameter **gerg_red_129** = gerg_mix_reducing(ident1 = "NC5", ident2 = "NC9", beta_v = 1., gamma_v = 1.034910633, beta_T = 1., gamma_T = 1.103421755)
- type(gerg_mix_reducing), parameter **gerg_red_130** = gerg_mix_reducing(ident1 = "NC4", ident2 = "NC6", beta_v = 1., gamma_v = 1.034995284, beta_T = 1., gamma_T = 1.00915706)
- type(gerg_mix_reducing), parameter **gerg_red_131** = gerg_mix_reducing(ident1 = "IC4", ident2 = "O2", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_132** = gerg_mix_reducing(ident1 = "NC10", ident2 = "AR", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_133** = gerg_mix_reducing(ident1 = "H2", ident2 = "O2", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)

- type(gerg_mix_reducing), parameter **gerg_red_134** = gerg_mix_reducing(ident1 = "IC5", ident2 = "NC8", beta_v = 1., gamma_v = 1.017880545, beta_T = 1., gamma_T = 1.00564748)
- type(gerg_mix_reducing), parameter **gerg_red_135** = gerg_mix_reducing(ident1 = "H2", ident2 = "H2O", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_136** = gerg_mix_reducing(ident1 = "H2S", ident2 = "AR", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_137** = gerg_mix_reducing(ident1 = "C2", ident2 = "H2S", beta_v = 1.010817909, gamma_v = 1.030988277, beta_T = 0.990197354, gamma_T = 0.90273666)
- type(gerg_mix_reducing), parameter **gerg_red_138** = gerg_mix_reducing(ident1 = "IC4", ident2 = "HE", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_139** = gerg_mix_reducing(ident1 = "NC6", ident2 = "CO", beta_v = 1., gamma_v = 1.155145836, beta_T = 1., gamma_T = 1.233272781)
- type(gerg_mix_reducing), parameter **gerg_red_140** = gerg_mix_reducing(ident1 = "NC6", ident2 = "NC9", beta_v = 1., gamma_v = 1.02076168, beta_T = 1., gamma_T = 1.055369591)
- type(gerg_mix_reducing), parameter **gerg_red_141** = gerg_mix_reducing(ident1 = "C1", ident2 = "C2", beta_v = 0.997547866, gamma_v = 1.006617867, beta_T = 0.996336508, gamma_T = 1.049707697)
- type(gerg_mix_reducing), parameter **gerg_red_142** = gerg_mix_reducing(ident1 = "NC4", ident2 = "NC9", beta_v = 1., gamma_v = 1.049219137, beta_T = 1., gamma_T = 1.014096448)
- type(gerg_mix_reducing), parameter **gerg_red_143** = gerg_mix_reducing(ident1 = "C1", ident2 = "NC9", beta_v = 1.002852287, gamma_v = 1.141895355, beta_T = 0.947716769, gamma_T = 1.528532478)
- type(gerg_mix_reducing), parameter **gerg_red_144** = gerg_mix_reducing(ident1 = "C1", ident2 = "NC6", beta_v = 0.958015294, gamma_v = 1.052643846, beta_T = 0.981844797, gamma_T = 1.330570181)
- type(gerg_mix_reducing), parameter **gerg_red_145** = gerg_mix_reducing(ident1 = "NC9", ident2 = "NC10", beta_v = 1., gamma_v = 1.00081052, beta_T = 1., gamma_T = 1.000182392)
- type(gerg_mix_reducing), parameter **gerg_red_146** = gerg_mix_reducing(ident1 = "NC4", ident2 = "H2S", beta_v = 0.908113163, gamma_v = 1.033366041, beta_T = 0.985962886, gamma_T = 0.926156602)
- type(gerg_mix_reducing), parameter **gerg_red_147** = gerg_mix_reducing(ident1 = "O2", ident2 = "H2S", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_148** = gerg_mix_reducing(ident1 = "NC5", ident2 = "AR", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_149** = gerg_mix_reducing(ident1 = "NC8", ident2 = "H2S", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_150** = gerg_mix_reducing(ident1 = "CO2", ident2 = "CO", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_151** = gerg_mix_reducing(ident1 = "IC4", ident2 = "AR", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_152** = gerg_mix_reducing(ident1 = "N2", ident2 = "H2O", beta_v = 1., gamma_v = 1.094749685, beta_T = 1., gamma_T = 0.968808467)
- type(gerg_mix_reducing), parameter **gerg_red_153** = gerg_mix_reducing(ident1 = "C2", ident2 = "NC10", beta_v = 0.995676258, gamma_v = 1.098361281, beta_T = 0.970918061, gamma_T = 1.237191558)
- type(gerg_mix_reducing), parameter **gerg_red_154** = gerg_mix_reducing(ident1 = "NC6", ident2 = "HE", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_155** = gerg_mix_reducing(ident1 = "NC9", ident2 = "H2S", beta_v = 1., gamma_v = 1.082905109, beta_T = 1., gamma_T = 1.086557826)
- type(gerg_mix_reducing), parameter **gerg_red_156** = gerg_mix_reducing(ident1 = "C3", ident2 = "IC5", beta_v = 1.040459289, gamma_v = 0.999432118, beta_T = 0.994364425, gamma_T = 1.0032695)
- type(gerg_mix_reducing), parameter **gerg_red_157** = gerg_mix_reducing(ident1 = "CO", ident2 = "AR", beta_v = 1., gamma_v = 1.159720623, beta_T = 1., gamma_T = 0.954215746)
- type(gerg_mix_reducing), parameter **gerg_red_158** = gerg_mix_reducing(ident1 = "NC6", ident2 = "H2S", beta_v = 0.754473958, gamma_v = 1.339283552, beta_T = 0.985891113, gamma_T = 0.956075596)
- type(gerg_mix_reducing), parameter **gerg_red_159** = gerg_mix_reducing(ident1 = "C2", ident2 = "NC8", beta_v = 1.007469726, gamma_v = 1.071917985, beta_T = 0.984068272, gamma_T = 1.168636194)
- type(gerg_mix_reducing), parameter **gerg_red_160** = gerg_mix_reducing(ident1 = "C3", ident2 = "NC7", beta_v = 1., gamma_v = 1.079648053, beta_T = 1., gamma_T = 1.050044169)
- type(gerg_mix_reducing), parameter **gerg_red_161** = gerg_mix_reducing(ident1 = "IC4", ident2 = "NC6", beta_v = 1., gamma_v = 1.010493989, beta_T = 1., gamma_T = 1.006018054)

- type(gerg_mix_reducing), parameter **gerg_red_162** = gerg_mix_reducing(ident1 = "CO2", ident2 = "HE", beta_v = 0.846647561, gamma_v = 0.864141549, beta_T = 0.76837763, gamma_T = 3.207456948)
- type(gerg_mix_reducing), parameter **gerg_red_163** = gerg_mix_reducing(ident1 = "IC4", ident2 = "CO", beta_v = 1., gamma_v = 1.087272232, beta_T = 1., gamma_T = 1.161390082)
- type(gerg_mix_reducing), parameter **gerg_red_164** = gerg_mix_reducing(ident1 = "N2", ident2 = "NC9", beta_v = 1., gamma_v = 1.100405929, beta_T = 0.95637945, gamma_T = 1.749119996)
- type(gerg_mix_reducing), parameter **gerg_red_165** = gerg_mix_reducing(ident1 = "C3", ident2 = "IC4", beta_v = 0.999243146, gamma_v = 1.001156119, beta_T = 0.998012298, gamma_T = 1.005250774)
- type(gerg_mix_reducing), parameter **gerg_red_166** = gerg_mix_reducing(ident1 = "IC5", ident2 = "CO", beta_v = 1., gamma_v = 1.116694577, beta_T = 1., gamma_T = 1.199326059)
- type(gerg_mix_reducing), parameter **gerg_red_167** = gerg_mix_reducing(ident1 = "NC7", ident2 = "HE", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_168** = gerg_mix_reducing(ident1 = "C3", ident2 = "O2", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_169** = gerg_mix_reducing(ident1 = "C3", ident2 = "NC6", beta_v = 1., gamma_v = 1.057872566, beta_T = 1., gamma_T = 1.025657518)
- type(gerg_mix_reducing), parameter **gerg_red_170** = gerg_mix_reducing(ident1 = "N2", ident2 = "HE", beta_v = 0.969501055, gamma_v = 0.932629867, beta_T = 0.692868765, gamma_T = 1.47183158)
- type(gerg_mix_reducing), parameter **gerg_red_171** = gerg_mix_reducing(ident1 = "NC8", ident2 = "H2O", beta_v = 1., gamma_v = 0.599484191, beta_T = 1., gamma_T = 0.662072469)
- type(gerg_mix_reducing), parameter **gerg_red_172** = gerg_mix_reducing(ident1 = "IC4", ident2 = "NC5", beta_v = 1., gamma_v = 1.002779804, beta_T = 1., gamma_T = 1.002495889)
- type(gerg_mix_reducing), parameter **gerg_red_173** = gerg_mix_reducing(ident1 = "C1", ident2 = "H2", beta_v = 1., gamma_v = 1.018702573, beta_T = 1., gamma_T = 1.352643115)
- type(gerg_mix_reducing), parameter **gerg_red_174** = gerg_mix_reducing(ident1 = "NC4", ident2 = "O2", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_175** = gerg_mix_reducing(ident1 = "H2", ident2 = "HE", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_176** = gerg_mix_reducing(ident1 = "N2", ident2 = "C3", beta_v = 0.974424681, gamma_v = 1.081025408, beta_T = 1.002677329, gamma_T = 1.201264026)
- type(gerg_mix_reducing), parameter **gerg_red_177** = gerg_mix_reducing(ident1 = "CO2", ident2 = "IC4", beta_v = 1.076551882, gamma_v = 1.081909003, beta_T = 1.023339824, gamma_T = 0.929982936)
- type(gerg_mix_reducing), parameter **gerg_red_178** = gerg_mix_reducing(ident1 = "NC8", ident2 = "O2", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_179** = gerg_mix_reducing(ident1 = "C3", ident2 = "CO", beta_v = 1., gamma_v = 1.108143673, beta_T = 1., gamma_T = 1.197564208)
- type(gerg_mix_reducing), parameter **gerg_red_180** = gerg_mix_reducing(ident1 = "NC7", ident2 = "NC9", beta_v = 1., gamma_v = 1.001370076, beta_T = 1., gamma_T = 1.001150096)
- type(gerg_mix_reducing), parameter **gerg_red_181** = gerg_mix_reducing(ident1 = "CO2", ident2 = "O2", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_182** = gerg_mix_reducing(ident1 = "IC5", ident2 = "NC6", beta_v = 1., gamma_v = 1.002995876, beta_T = 1., gamma_T = 1.001204174)
- type(gerg_mix_reducing), parameter **gerg_red_183** = gerg_mix_reducing(ident1 = "C3", ident2 = "NC4", beta_v = 0.999795868, gamma_v = 1.003264179, beta_T = 1.000310289, gamma_T = 1.007392782)
- type(gerg_mix_reducing), parameter **gerg_red_184** = gerg_mix_reducing(ident1 = "N2", ident2 = "NC4", beta_v = 0.99608261, gamma_v = 1.146949309, beta_T = 0.994515234, gamma_T = 1.304886838)
- type(gerg_mix_reducing), parameter **gerg_red_185** = gerg_mix_reducing(ident1 = "N2", ident2 = "H2", beta_v = 0.972532065, gamma_v = 0.970115357, beta_T = 0.946134337, gamma_T = 1.175696583)
- type(gerg_mix_reducing), parameter **gerg_red_186** = gerg_mix_reducing(ident1 = "C1", ident2 = "IC4", beta_v = 1.011240388, gamma_v = 1.054319053, beta_T = 0.980315756, gamma_T = 1.161117729)
- type(gerg_mix_reducing), parameter **gerg_red_187** = gerg_mix_reducing(ident1 = "CO2", ident2 = "C3", beta_v = 0.996898004, gamma_v = 1.047596298, beta_T = 1.033620538, gamma_T = 0.908772477)
- type(gerg_mix_reducing), parameter **gerg_red_188** = gerg_mix_reducing(ident1 = "C2", ident2 = "H2O", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_189** = gerg_mix_reducing(ident1 = "NC8", ident2 = "AR", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)

- type(gerg_mix_reducing), parameter **gerg_red_190** = gerg_mix_reducing(ident1 = "NC7", ident2 = "O2", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_191** = gerg_mix_reducing(ident1 = "IC5", ident2 = "O2", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_192** = gerg_mix_reducing(ident1 = "C1", ident2 = "CO2", beta_v = 0.999518072, gamma_v = 1.002806594, beta_T = 1.02262449, gamma_T = 0.975665369)
- type(gerg_mix_reducing), parameter **gerg_red_193** = gerg_mix_reducing(ident1 = "NC7", ident2 = "H2O", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_194** = gerg_mix_reducing(ident1 = "NC8", ident2 = "NC10", beta_v = 1., gamma_v = 1.002553544, beta_T = 1., gamma_T = 1.007186267)
- type(gerg_mix_reducing), parameter **gerg_red_195** = gerg_mix_reducing(ident1 = "H2S", ident2 = "HE", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_196** = gerg_mix_reducing(ident1 = "NC8", ident2 = "H2", beta_v = 1., gamma_v = 1.305249405, beta_T = 1., gamma_T = 2.191555216)
- type(gerg_mix_reducing), parameter **gerg_red_197** = gerg_mix_reducing(ident1 = "CO2", ident2 = "C2", beta_v = 1.002525718, gamma_v = 1.032876701, beta_T = 1.013871147, gamma_T = 0.90094953)
- type(gerg_mix_reducing), parameter **gerg_red_198** = gerg_mix_reducing(ident1 = "NC5", ident2 = "HE", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_199** = gerg_mix_reducing(ident1 = "IC5", ident2 = "NC7", beta_v = 1., gamma_v = 1.009928206, beta_T = 1., gamma_T = 1.003194615)
- type(gerg_mix_reducing), parameter **gerg_red_200** = gerg_mix_reducing(ident1 = "N2", ident2 = "IC5", beta_v = 1., gamma_v = 1.154135439, beta_T = 1., gamma_T = 1.38177077)
- type(gerg_mix_reducing), parameter **gerg_red_201** = gerg_mix_reducing(ident1 = "NC5", ident2 = "NC8", beta_v = 1., gamma_v = 1.069223964, beta_T = 1., gamma_T = 1.016422347)
- type(gerg_mix_reducing), parameter **gerg_red_202** = gerg_mix_reducing(ident1 = "C1", ident2 = "CO", beta_v = 0.997340772, gamma_v = 1.006102927, beta_T = 0.987411732, gamma_T = 0.987473033)
- type(gerg_mix_reducing), parameter **gerg_red_203** = gerg_mix_reducing(ident1 = "CO2", ident2 = "H2O", beta_v = 0.949055959, gamma_v = 1.542328793, beta_T = 0.997372205, gamma_T = 0.775453996)
- type(gerg_mix_reducing), parameter **gerg_red_204** = gerg_mix_reducing(ident1 = "CO2", ident2 = "IC5", beta_v = 1.060793104, gamma_v = 1.116793198, beta_T = 1.019180957, gamma_T = 0.961218039)
- type(gerg_mix_reducing), parameter **gerg_red_205** = gerg_mix_reducing(ident1 = "NC5", ident2 = "H2S", beta_v = 0.984613203, gamma_v = 1.076539234, beta_T = 0.962006651, gamma_T = 0.959065662)
- type(gerg_mix_reducing), parameter **gerg_red_206** = gerg_mix_reducing(ident1 = "CO", ident2 = "H2S", beta_v = 0.795660392, gamma_v = 1.101731308, beta_T = 1.025536736, gamma_T = 1.022749748)
- type(gerg_mix_reducing), parameter **gerg_red_207** = gerg_mix_reducing(ident1 = "CO2", ident2 = "NC4", beta_v = 1.174760923, gamma_v = 1.222437324, beta_T = 1.018171004, gamma_T = 0.911498231)
- type(gerg_mix_reducing), parameter **gerg_red_208** = gerg_mix_reducing(ident1 = "IC4", ident2 = "H2O", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(gerg_mix_reducing), parameter **gerg_red_209** = gerg_mix_reducing(ident1 = "C2", ident2 = "CO", beta_v = 1., gamma_v = 1.201417898, beta_T = 1., gamma_T = 1.069224728)
- type(gerg_mix_reducing), parameter **gerg_red_210** = gerg_mix_reducing(ident1 = "NC8", ident2 = "HE", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- integer, parameter **max_gerg_mix_reducing** = 210
- type(gerg_mix_reducing), dimension(max_gerg_mix_reducing), parameter **gerg_mix_reducingdb** = (/ gerg_red_1,gerg_red_2,gerg_red_3,gerg_red_4,gerg_red_5,gerg_red_6, gerg_red_7,gerg_red_8,gerg_↩ red_9,gerg_red_10,gerg_red_11,gerg_red_12, gerg_red_13,gerg_red_14,gerg_red_15,gerg_red_16,gerg↩ _red_17,gerg_red_18, gerg_red_19,gerg_red_20,gerg_red_21,gerg_red_22,gerg_red_23,gerg_red_24, gerg_red_25,gerg_red_26,gerg_red_27,gerg_red_28,gerg_red_29,gerg_red_30, gerg_red_31,gerg_red↩ _32,gerg_red_33,gerg_red_34,gerg_red_35,gerg_red_36, gerg_red_37,gerg_red_38,gerg_red_39,gerg↩ _red_40,gerg_red_41,gerg_red_42, gerg_red_43,gerg_red_44,gerg_red_45,gerg_red_46,gerg_red_↩ 47,gerg_red_48, gerg_red_49,gerg_red_50,gerg_red_51,gerg_red_52,gerg_red_53,gerg_red_54, gerg↩ _red_55,gerg_red_56,gerg_red_57,gerg_red_58,gerg_red_59,gerg_red_60, gerg_red_61,gerg_red_↩ 62,gerg_red_63,gerg_red_64,gerg_red_65,gerg_red_66, gerg_red_67,gerg_red_68,gerg_red_69,gerg↩ _red_70,gerg_red_71,gerg_red_72, gerg_red_73,gerg_red_74,gerg_red_75,gerg_red_76,gerg_red_↩ 77,gerg_red_78, gerg_red_79,gerg_red_80,gerg_red_81,gerg_red_82,gerg_red_83,gerg_red_84, gerg↩ _red_85,gerg_red_86,gerg_red_87,gerg_red_88,gerg_red_89,gerg_red_90, gerg_red_91,gerg_red_↩ 92,gerg_red_93,gerg_red_94,gerg_red_95,gerg_red_96, gerg_red_97,gerg_red_98,gerg_red_99,gerg_↩

red_100,gerg_red_101,gerg_red_102, gerg_red_103,gerg_red_104,gerg_red_105,gerg_red_106,gerg_↩
red_107,gerg_red_108, gerg_red_109,gerg_red_110,gerg_red_111,gerg_red_112,gerg_red_113,gerg_↩
red_114, gerg_red_115,gerg_red_116,gerg_red_117,gerg_red_118,gerg_red_119,gerg_red_120, gerg_↩
red_121,gerg_red_122,gerg_red_123,gerg_red_124,gerg_red_125,gerg_red_126, gerg_red_127,gerg_↩
red_128,gerg_red_129,gerg_red_130,gerg_red_131,gerg_red_132, gerg_red_133,gerg_red_134,gerg_↩
red_135,gerg_red_136,gerg_red_137,gerg_red_138, gerg_red_139,gerg_red_140,gerg_red_141,gerg_↩
red_142,gerg_red_143,gerg_red_144, gerg_red_145,gerg_red_146,gerg_red_147,gerg_red_148,gerg_↩
red_149,gerg_red_150, gerg_red_151,gerg_red_152,gerg_red_153,gerg_red_154,gerg_red_155,gerg_↩
red_156, gerg_red_157,gerg_red_158,gerg_red_159,gerg_red_160,gerg_red_161,gerg_red_162, gerg_↩
red_163,gerg_red_164,gerg_red_165,gerg_red_166,gerg_red_167,gerg_red_168, gerg_red_169,gerg_↩
red_170,gerg_red_171,gerg_red_172,gerg_red_173,gerg_red_174, gerg_red_175,gerg_red_176,gerg_↩
red_177,gerg_red_178,gerg_red_179,gerg_red_180, gerg_red_181,gerg_red_182,gerg_red_183,gerg_↩
red_184,gerg_red_185,gerg_red_186, gerg_red_187,gerg_red_188,gerg_red_189,gerg_red_190,gerg_↩
red_191,gerg_red_192, gerg_red_193,gerg_red_194,gerg_red_195,gerg_red_196,gerg_red_197,gerg_↩
red_198, gerg_red_199,gerg_red_200,gerg_red_201,gerg_red_202,gerg_red_203,gerg_red_204, gerg_↩
red_205,gerg_red_206,gerg_red_207,gerg_red_208,gerg_red_209,gerg_red_210 /)

- type(gerg_mix_data), parameter **gerg_mix1** = gerg_mix_data(ident1 = "C2", ident2 = "C3", Fij = 0.13042476515, num_mix = 10, n_mix = (/ 0.25574776844118d1,-0.79846357136353d1,0.↩
47859131465806d1, -0.73265392369587,0.13805471345312d1,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.55527385721943d-4,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), eta_mix = (/ 0.0,0.↩
0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), beta_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), num_exp = 0)
- type(gerg_mix_data), parameter **gerg_mix2** = gerg_mix_data(ident1 = "C2", ident2 = "IC4", Fij = 0.260632376098, num_mix = 10, n_mix = (/ 0.25574776844118d1,-0.79846357136353d1,0.↩
47859131465806d1, -0.73265392369587,0.13805471345312d1,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.55527385721943d-4,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), eta_mix = (/ 0.0,0.↩
0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), beta_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), num_exp = 0)
- type(gerg_mix_data), parameter **gerg_mix3** = gerg_mix_data(ident1 = "C1", ident2 = "NC4", Fij = 1., num_mix = 10, n_mix = (/ 0.25574776844118d1,-0.79846357136353d1,0.47859131465806d1, -0.↩
73265392369587,0.13805471345312d1,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩
11836314681968, 0.55527385721943d-4,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), eta_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), beta_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), num_exp = 0)
- type(gerg_mix_data), parameter **gerg_mix4** = gerg_mix_data(ident1 = "C1", ident2 = "N2", Fij = 1., num_mix = 9, n_mix = (/ -0.98038985517335d-2,0.42487270143005d-3,-0.34800214576142d-1, -0.↩
13333813013896,-0.11993694974627d-1,0.69243379775168d-1, -0.31022508148249,0.24495491753226,0.↩
22369816716981, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 0.0,1.85,7.85, 5.4,0.0,0.75, 2.8,4.45,4.25, 0.0d0,0.0d0,0.↩
0d0 /), d_mix = (/ 1,4,1,2,2,2, 2,2,3,0,0,0 /), eta_mix = (/ 0.0,0.0,1., 1.,0.25,0., 0.,0.,0., 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0,0.0,0.5, 0.5,0.5,0.5, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0,0.0,0.5, 0.5,0.5,0.5, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0,0.0,1., 1.,2.5,3., 3.,3.,3., 0.0d0,0.0d0,0.0d0 /), num_exp = 7)
- type(gerg_mix_data), parameter **gerg_mix5** = gerg_mix_data(ident1 = "NC4", ident2 = "IC4", Fij = -0.0551240293009, num_mix = 10, n_mix = (/ 0.25574776844118d1,-0.79846357136353d1,0.↩
47859131465806d1, -0.73265392369587,0.13805471345312d1,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.55527385721943d-4,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), eta_mix = (/ 0.0,0.↩
0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), beta_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), num_exp = 0)
- type(gerg_mix_data), parameter **gerg_mix6** = gerg_mix_data(ident1 = "N2", ident2 = "C2", Fij = 1., num_↩
mix = 6, n_mix = (/ -0.47376518126608,0.48961193461001,-0.57011062090535d-2, -0.19966820041320,-

0.69411103101723,0.69226192739021, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 0.0,0.05,0.↩
0, 3.65,4.9,4.45, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 2,2,3,1,2,2, 0,0,0,0,0,0 /), eta_mix = (/ 0.0,0.0,0.0, 1.,1.,0.875, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0,0.0,0.0, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0,0.0,0.0, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0,0.0,0.0, 1.,1.,1.25, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 3)

- type(gerg_mix_data), parameter **gerg_mix7** = gerg_mix_data(ident1 = "C2", ident2 = "NC4", Fij = 0.281570073085, num_mix = 10, n_mix = (/ 0.25574776844118d1,-0.79846357136353d1,0.↩
47859131465806d1, -0.73265392369587,0.13805471345312d1,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.55527385721943d-4,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), eta_mix = (/ 0.0,0.↩
0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), beta_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), num_exp = 0)
- type(gerg_mix_data), parameter **gerg_mix8** = gerg_mix_data(ident1 = "N2", ident2 = "CO2", Fij = 1., num_↩
mix = 6, n_mix = (/ 0.28661625028399,-0.10919833861247,-0.11374032082270d1, 0.76580544237358,0.↩
42638000926819d-2,0.17673538204534, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 1.85,1.4,3.↩
2, 2.5,8.0,3.75, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 2,3,1,1,1,2, 0,0,0,0,0,0 /), eta_mix = (/ 0.0,0.0,0.25, 0.25,0.,0., 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0,0.0,0.5, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0,0.0,0.5, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0,0.0,0.75, 1.,2.,3., 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 4)
- type(gerg_mix_data), parameter **gerg_mix9** = gerg_mix_data(ident1 = "C1", ident2 = "C3", Fij = 1., num_mix = 9, n_mix = (/ 0.13746429958576d-1,-0.74425012129552d-2,-0.45516600213685d-2, -0.↩
54546603350237d-2,0.23682016824471d-2,0.18007763721438, -0.44773942932486,0.19327374888200d-1,-0.30632197804624, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 1.85,3.95,0.0, 1.85,3.85,5.25, 3.85,0.2,6.5, 0.↩
0d0,0.0d0,0.0d0 /), d_mix = (/ 3,3,4,4,4,1, 1,1,2,0,0,0 /), eta_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.25, 0.25,0.,0., 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.5, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.5, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.75, 1.,2.,3., 0.0d0,0.0d0,0.0d0 /), num_exp = 4)
- type(gerg_mix_data), parameter **gerg_mix10** = gerg_mix_data(ident1 = "C1", ident2 = "C2", Fij = 1., num_mix = 12, n_mix = (/ -0.80926050298746d-3,-0.75381925080059d-3,-0.41618768891219d-1, -0.↩
23452173681569,0.14003840584586,0.63281744807738d-1, -0.34660425848809d-1,-0.23918747334251,0.↩
19855255066891d-2, 0.61777746171555d1,-0.69575358271105d1,0.10630185306388d1 /), t_mix = (/ 0.↩
65,1.55,3.1, 5.9,7.05,3.35, 1.2,5.8,2.7, 0.45,0.55,1.95 /), d_mix = (/ 3,4,1,2,2,2, 2,2,2,3,3,3 /), eta_mix = (/ 0.0,0.0,1., 1.,1.,0.875, 0.75,0.5,0., 0.,0.,0. /), gamma_mix = (/ 0.0,0.0,0.5, 0.5,0.5,0.5, 0.5,0.5,0.5, 0.5,0.5,0.5 /), epsilon_mix = (/ 0.0,0.0,0.5, 0.5,0.5,0.5, 0.5,0.5,0.5, 0.5,0.5,0.5 /), beta_mix = (/ 0.0,0.0,1., 1.,1.,1.25, 1.5,2.,3., 3.,3.,3. /), num_exp = 10)
- type(gerg_mix_data), parameter **gerg_mix11** = gerg_mix_data(ident1 = "C3", ident2 = "IC4", Fij = -0.0551609771024, num_mix = 10, n_mix = (/ 0.25574776844118d1,-0.79846357136353d1,0.↩
47859131465806d1, -0.73265392369587,0.13805471345312d1,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.55527385721943d-4,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), eta_mix = (/ 0.0,0.↩
0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), beta_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), num_exp = 0)
- type(gerg_mix_data), parameter **gerg_mix12** = gerg_mix_data(ident1 = "C1", ident2 = "H2", Fij = 1., num_mix = 4, n_mix = (/ -0.25157134971934,-0.62203841111983d-2,0.88850315184396d-1, -0.↩
35592212573239d-1,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 2.0,-1.0,1.75, 1.↩
4,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,3,3,4,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0,0.0,0.0, 0.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0,0.0,0.↩
0, 0.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0,0.0,0.0, 0.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0,0.0,0.0, 0.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0)
- type(gerg_mix_data), parameter **gerg_mix13** = gerg_mix_data(ident1 = "C3", ident2 = "NC4", Fij = 0.0312572600489, num_mix = 10, n_mix = (/ 0.25574776844118d1,-0.79846357136353d1,0.↩
47859131465806d1, -0.73265392369587,0.13805471345312d1,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.55527385721943d-4,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7,

0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), eta_mix = (/ 0.0,0.↩
0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0,
0.0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), beta_mix = (/
0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), num_exp = 0)

- type(gerg_mix_data), parameter **gerg_mix14** = gerg_mix_data(ident1 = "C1", ident2 = "IC4", Fij = 0.771035405688, num_mix = 10, n_mix = (/ 0.25574776844118d1,-0.79846357136353d1,0.↩
47859131465806d1, -0.73265392369587,0.13805471345312d1,0.28349603476365, -0.49087385940425,-
0.10291888921447,0.11836314681968, 0.55527385721943d-4,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7,
0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), eta_mix = (/ 0.0,0.↩
0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0,
0.0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), beta_mix = (/
0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0,0.0, 0.0,0.0d0,0.0d0 /), num_exp = 0)
- type(gerg_mix_data), parameter **gerg_mix15** = gerg_mix_data(ident1 = "C1", ident2 = "CO2", Fij = 1., num_mix = 6, n_mix = (/ -0.10859387354942,0.80228576727389d-1,-0.93303985115717d-2, 0.↩
40989274005848d-1,-0.24338019772494,0.23855347281124, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /),
t_mix = (/ 2.6,1.95,0.0, 3.95,7.95,8.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,2,3,1,2,3,
0,0,0,0,0,0 /), eta_mix = (/ 0.0,0.0,0.0, 1.,0.5,0., 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/
0.0,0.0,0.0, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0,0.0,0.0, 0.5,0.5,0.5,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0,0.0,0.0, 1.,2.,3., 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), num_exp = 3)
- integer, parameter **max_gerg_mix_data** = 15
- type(gerg_mix_data), dimension(max_gerg_mix_data), parameter **gerg_mix_datadb** = (/ gerg_mix1,gerg↩
_mix2,gerg_mix3,gerg_mix4,gerg_mix5,gerg_mix6, gerg_mix7,gerg_mix8,gerg_mix9,gerg_mix10,gerg_↩
mix11,gerg_mix12, gerg_mix13,gerg_mix14,gerg_mix15 /)

5.25.1 Detailed Description

Automatically generated file gergmixdb.f90 Time stamp: 2022-10-03T21:42:11.071422.

5.26 h2o_gibbs Module Reference

Module calculating thermodynamic potentials and differentials for solid H2O.

Functions/Subroutines

- subroutine, public sho2_init (sl, gl)

  Set reference potentials (H2O) Input: Liquid entropy and Gibbs energy evaluated at (T0_H2O,P0_H2O)
- real function, public sh2o_gibbs (t, p)

  Gibbs free energy for ice (H2O) (J/mol)
- real function, public sh2o_dgdt (t, p)

  Gibbs free energy for ice (H2O) - Differential wrpt. temperature at constant pressure (J/mol/K)
- real function, public sh2o_d2gdt2 (t, p)

  Gibbs free energy for ice (H2O) - Second differential wrpt. temperature at constant pressure (J/mol/K)
- real function, public sh2o_d2gdtdp (t, p)

  Gibbs free energy for ice (H2O) - Second differential wrpt. temperature and pressure (m3/mol/K)
- real function, public sh2o_dgdp (t, p)

  Gibbs free energy for ice (H2O) - Differential wrpt. pressure at constant temperature (m3/mol)
- real function, public sh2o_d2gdp2 (t, p)

  Gibbs free energy for ice (H2O) - Second differential wrpt. pressure at constant temperature (m3/mol/Pa)
- real function, public sh2o_specvol (t, p)

  Specific volume for ice (H2O) (m3/mol)
- real function, public sh2o_entropy (t, p)

  Ice (H2O) entropy (J/mol/K)
- real function, public sh2o_enthalpy (t, p)

Ice (H2O) enthalpy (J/mol)

- real function, public sh2o_energy (t, p)

  Ice (H2O) energy (J/mol)
- real function, public sh2o_helmholtz (t, p)

  Ice (H2O) helmholtz energy (J/mol)
- real function, public sh2o_heat_capacity (t, p)

  Ice (H2O) heat capacity at constant pressure (J/mol/K)

Variables

- real, parameter, public **p0_h2o** = 101325.0

  Pa.
- real, parameter, public **t0_h2o** = 273.152519

  K.
- real, parameter, public **dsmelt** = mw_H2O∗333.426516∗1.0e3/T0_H2O

  J/mol/K - (Sliq - Ssolid)

5.26.1 Detailed Description

Module calculating thermodynamic potentials and differentials for solid H2O.
Ref: A New Equation of State for H$_2$O Ice Ih Rainer Feistel and Wolfgang Wagner Journal of Physical and Chemical Reference Data (J. Phys. Chem. Ref. Data) Pages: 1021-1047 January 2006 Volume(Number): 35(2) doi: http://dx.doi.org/10.1063/1.2183324

5.26.2 Function/Subroutine Documentation

5.26.2.1 sh2o_d2gdp2()

```
real function, public h2o_gibbs::sh2o_d2gdp2 (
            real, intent(in) t,
            real, intent(in) p )
```

Gibbs free energy for ice (H2O) - Second differential wrpt. pressure at constant temperature (m3/mol/Pa)

Parameters

in	*t*	K
in	*p*	Pa

Returns

m3/mol/Pa

5.26.2.2 sh2o_d2gdt2()

```
real function, public h2o_gibbs::sh2o_d2gdt2 (
            real, intent(in) t,
            real, intent(in) p )
```

Gibbs free energy for ice (H2O) - Second differential wrpt. temperature at constant pressure (J/mol/K)

Parameters

in	*t*	K
in	*p*	Pa

Returns

J/mol/K2

5.26.2.3 sh2o_d2gdtdp()

```
real function, public h2o_gibbs::sh2o_d2gdtdp (
            real, intent(in) t,
            real, intent(in) p )
```

Gibbs free energy for ice (H2O) - Second differential wrpt. temperature and pressure (m3/mol/K)

Parameters

in	*t*	K
in	*p*	Pa

Returns

m3/mol/K

5.26.2.4 sh2o_dgdp()

```
real function, public h2o_gibbs::sh2o_dgdp (
            real, intent(in) t,
            real, intent(in) p )
```

Gibbs free energy for ice (H2O) - Differential wrpt. pressure at constant temperature (m3/mol)

Parameters

in	*t*	K
in	*p*	Pa

Returns

m3/mol

5.26.2.5 sh2o_dgdt()

```
real function, public h2o_gibbs::sh2o_dgdt (
            real, intent(in) t,
            real, intent(in) p )
```

Gibbs free energy for ice (H2O) - Differential wrpt. temperature at constant pressure (J/mol/K)

Parameters

in	*t*	K
in	*p*	Pa

Returns

J/mol/K

5.26.2.6 sh2o_energy()

```
real function, public h2o_gibbs::sh2o_energy (
```

```
            real, intent(in) t,
            real, intent(in) p )
```

Ice (H2O) energy (J/mol)

Parameters

in	*t*	K
in	*p*	Pa

Returns

J/mol

5.26.2.7 sh2o_enthalpy()

```
real function, public h2o_gibbs::sh2o_enthalpy (
            real, intent(in) t,
            real, intent(in) p )
```

Ice (H2O) enthalpy (J/mol)

Parameters

in	*t*	K
in	*p*	Pa

5.26.2.8 sh2o_entropy()

```
real function, public h2o_gibbs::sh2o_entropy (
            real, intent(in) t,
            real, intent(in) p )
```

Ice (H2O) entropy (J/mol/K)

Parameters

in	*t*	K
in	*p*	Pa

Returns

J/mol/K

5.26.2.9 sh2o_gibbs()

```
real function, public h2o_gibbs::sh2o_gibbs (
            real, intent(in) t,
            real, intent(in) p )
```

Gibbs free energy for ice (H2O) (J/mol)

Parameters

in	*t*	K
in	*p*	Pa

Returns

J/mol

5.26.2.10 sh2o_heat_capacity()

```
real function, public h2o_gibbs::sh2o_heat_capacity (
            real, intent(in) t,
            real, intent(in) p )
```

Ice (H2O) heat capacity at constant pressure (J/mol/K)

Parameters

in	*t*	K
in	*p*	Pa

5.26.2.11 sh2o_helmholtz()

```
real function, public h2o_gibbs::sh2o_helmholtz (
            real, intent(in) t,
            real, intent(in) p )
```

Ice (H2O) helmholtz energy (J/mol)

Parameters

in	*t*	K
in	*p*	Pa

Returns

J/mol

5.26.2.12 sh2o_specvol()

```
real function, public h2o_gibbs::sh2o_specvol (
            real, intent(in) t,
            real, intent(in) p )
```

Specific volume for ice (H2O) (m3/mol)

Parameters

in	*t*	K
in	*p*	Pa

Returns

m3/mol

5.26.2.13 sho2_init()

```
subroutine, public h2o_gibbs::sho2_init (
            real, intent(in) sl,
            real, intent(in) gl )
```

Set reference potentials (H2O) Input: Liquid entropy and Gibbs energy evaluated at (T0_H2O,P0_H2O)

Parameters

in	*sl*	Entropy - J/mol/K
in	*gl*	Gibbs energy - J/mol

5.27 hyperdual_mod Module Reference

Hyperdual number definition & type declaration.

Data Types

- interface abs
- interface acos
- interface asin
- interface assignment(=)
- interface atan
- interface atan2
- interface cos
- interface cosh
- interface exp
- type hyperdual

  Derived type for hyperdual numbers.
- interface int
- interface log
- interface log10
- interface max
- interface min
- interface nint
- interface operator(∗)
- interface operator(∗∗)
- interface operator(+)
- interface operator(-)
- interface operator(.eq.)
- interface operator(.ge.)
- interface operator(.gt.)
- interface operator(.le.)
- interface operator(.lt.)
- interface operator(.ne.)
- interface operator(/)
- interface real
- interface sign
- interface sin
- interface sinh
- interface sqrt
- interface sum
- interface tan
- interface tanh

Functions/Subroutines

- elemental subroutine **equalhyperdualhyperdual** (res, inp)
- elemental subroutine **equalhyperdualreal** (res, inp)
- elemental type(hyperdual) function **plushyperdualhyperdual** (v1)
- elemental type(hyperdual) function **addhyperdualhyperdual** (v1, v2)
- elemental type(hyperdual) function **addhyperdualreal** (v1, v2)
- elemental type(hyperdual) function **addrealhyperdual** (v1, v2)
- elemental type(hyperdual) function **minushyperdualhyperdual** (v1)
- elemental type(hyperdual) function **subtracthyperdualhyperdual** (v1, v2)
- elemental type(hyperdual) function **subtracthyperdualreal** (v1, v2)
- elemental type(hyperdual) function **subtractrealhyperdual** (v1, v2)
- elemental type(hyperdual) function **multiplyhyperdualhyperdual** (v1, v2)
- elemental type(hyperdual) function **multiplyhyperdualreal** (v1, v2)
- elemental type(hyperdual) function **multiplyrealhyperdual** (v1, v2)
- elemental type(hyperdual) function **multiplyhyperdualint** (v1, v2)
- elemental type(hyperdual) function **multiplyinthyperdual** (v1, v2)
- elemental type(hyperdual) function **dividehyperdualhyperdual** (v1, v2)
- elemental type(hyperdual) function **dividehyperdualreal** (v1, v2)
- elemental type(hyperdual) function **dividerealhyperdual** (v1, v2)
- elemental type(hyperdual) function **powerhyperdualint** (v1, v2)
- elemental type(hyperdual) function **powerhyperdualhyperdual** (v1, v2)
- elemental type(hyperdual) function **powerhyperdualreal** (v1, v2)
- pure type(hyperdual) function **sumhyperdual** (v1, mask)
- pure type(hyperdual) function, dimension(:), allocatable **sumhyperdual2** (v1, dim)
- logical function **eq_dd** (lhs, rhs)
- logical function **eq_dr** (lhs, rhs)
- logical function **eq_rd** (lhs, rhs)
- logical function **eq_di** (lhs, rhs)
- logical function **eq_id** (lhs, rhs)
- logical function **ne_dd** (lhs, rhs)
- logical function **ne_dr** (lhs, rhs)
- logical function **ne_rd** (lhs, rhs)
- logical function **ne_di** (lhs, rhs)
- logical function **ne_id** (lhs, rhs)
- logical function **lt_dd** (lhs, rhs)
- logical function **lt_dr** (lhs, rhs)
- logical function **lt_rd** (lhs, rhs)
- logical function **lt_di** (lhs, rhs)
- logical function **lt_id** (lhs, rhs)
- logical function **le_dd** (lhs, rhs)
- logical function **le_dr** (lhs, rhs)
- logical function **le_rd** (lhs, rhs)
- logical function **le_di** (lhs, rhs)
- logical function **le_id** (lhs, rhs)
- logical function **gt_dd** (lhs, rhs)
- logical function **gt_dr** (lhs, rhs)
- logical function **gt_rd** (lhs, rhs)
- logical function **gt_di** (lhs, rhs)
- logical function **gt_id** (lhs, rhs)
- logical function **ge_dd** (lhs, rhs)
- logical function **ge_dr** (lhs, rhs)
- logical function **ge_rd** (lhs, rhs)
- logical function **ge_di** (lhs, rhs)
- logical function **ge_id** (lhs, rhs)

- elemental type(hyperdual) function **abshyperdual** (v1)
- elemental integer function **inthyperdual** (v1)
- elemental integer function **ninthyperdual** (v1)
- elemental real(dp) function **realhyperdual** (v1)
- elemental type(hyperdual) function **sign_dd** (v1, v2)
- elemental type(hyperdual) function **sign_dr** (v1, v2)
- elemental type(hyperdual) function **sign_rd** (v1, v2)
- elemental type(hyperdual) function **sinhyperdual** (v1)
- elemental type(hyperdual) function **coshyperdual** (v1)
- elemental type(hyperdual) function **tanhyperdual** (v1)
- elemental type(hyperdual) function **sqrthyperdual** (v1)
- elemental type(hyperdual) function **loghyperdual** (v1)
- elemental type(hyperdual) function **log10hyperdual** (v1)
- elemental type(hyperdual) function **exphyperdual** (v1)
- elemental type(hyperdual) function **sinhhyperdual** (v1)
- elemental type(hyperdual) function **coshhyperdual** (v1)
- elemental type(hyperdual) function **tanhhyperdual** (v1)
- elemental type(hyperdual) function **acoshyperdual** (v1)
- elemental type(hyperdual) function **asinhyperdual** (v1)
- elemental type(hyperdual) function **atanhyperdual** (v1)
- elemental type(hyperdual) function **atan2hyperdual** (v1, v2)
- elemental type(hyperdual) function **max_dd** (v1, v2)
- elemental type(hyperdual) function **max_ddd** (v1, v2, v3)
- elemental type(hyperdual) function **max_dr** (v1, v2)
- elemental type(hyperdual) function **max_rd** (v1, v2)
- elemental type(hyperdual) function **min_dd** (v1, v2)
- elemental type(hyperdual) function **min_dr** (v1, v2)
- elemental type(hyperdual) function **min_rd** (v1, v2)
- elemental logical function **hd_all_members_zero** (v1)

5.27.1 Detailed Description

Hyperdual number definition & type declaration.
Original code provided by Philipp Rehner and Gernot Bauer, Institute of Thermodynamics and Thermal Process Engineering (ITT), University of Stuttgart, Stuttgart, Germany
Extended to third order differentials April 2023, Morten Hammer, NTNU.

Hypderdual numbers

Hypderdual numbers extend the idea of additional, non-real components from one non-real component (complex numbers) to seven non-real components: ε_1, ε_2, ε_3, $\varepsilon_1\varepsilon_2$, $\varepsilon_1\varepsilon_3$, $\varepsilon_2\varepsilon_3$ and $\varepsilon_1\varepsilon_2\varepsilon_3$ Hyperdual numbers require : $(\varepsilon_1)^2 = 0$, $(\varepsilon_2)^2 = 0$ and $(\varepsilon_3)^2 = 0$. This leads to the fact, that the Taylor series of a function with hyperdual arguments can be truncated *exactly* after the third derivative term:

$$f(\mathbf{x}+h_1\varepsilon_1+h_2\varepsilon_2+h_1h_2\varepsilon_1\varepsilon_2) = f(\mathbf{x})+h_1f'(\mathbf{x})\varepsilon_1+h_2f'(\mathbf{x})\varepsilon_2+h_3f'(\mathbf{x})\varepsilon_3+h_1h_2f''(\mathbf{x})\varepsilon_1\varepsilon_2+h_1h_3f''(\mathbf{x})\varepsilon_1\varepsilon_3+h_2h_3f''(\mathbf{x})\varepsilon_2\varepsilon_3+h_1$$

Because there is *no truncation error*, all first, second and third order derivatives can be obtained *exactly*, regardless of the step size " h_1, and h_2 and h_3. The derivatives (exemplified by a second order case) can be obtained for a function $f(\mathbf{x})$ with multiple variables $\mathbf{x} \in \mathbb{R}^n$ via

$$\begin{aligned}
\frac{\partial f(\mathbf{x})}{\partial x_i} &= \frac{\varepsilon_{1,\mathrm{part}}\Big\{f(\mathbf{x}+h_1\varepsilon_1\mathbf{e}_i+h_2\varepsilon_2\mathbf{e}_j+h_1h_2\mathbf{0})\Big\}}{h_1} \\
\frac{\partial f(\mathbf{x})}{\partial x_i} &= \frac{\varepsilon_{2,\mathrm{part}}\Big\{f(\mathbf{x}+h_1\varepsilon_1\mathbf{e}_i+h_2\varepsilon_2\mathbf{e}_j+h_1h_2\mathbf{0})\Big\}}{h_2} \\
\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} &= \frac{(\varepsilon_1\varepsilon_2)_{\mathrm{part}}\Big\{f(\mathbf{x}+h_1\varepsilon_1\mathbf{e}_i+h_2\varepsilon_2\mathbf{e}_j+h_1h_2\mathbf{0})\Big\}}{h_1h_2}
\end{aligned}$$

where $\mathbf{e}_i$ and $\mathbf{e}_j$ are unit vectors, which are all zero except for the i-th and j-th component, respectively.

Computation principles for hypderdual numbers

Hyperdual numbers $\mathbf{x} \in \mathbb{HD}$ can be expressed as tuples: $\mathbf{x} = [x_0, x_1, x_2, x_3, x_{12}, x_{13}, x_{23}, x_{123}] = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + x_3\varepsilon_3 + x_{12}\varepsilon_1\varepsilon_2 + x_{13}\varepsilon_1\varepsilon_3 + x_{23}\varepsilon_2\varepsilon_3 + x_{123}\varepsilon_1\varepsilon_2\varepsilon_3$. By using the Taylor expansion of the function $f(\mathbf{x})$ one gets computation priniple for functions with hyperdual arguments from

$$f(\mathbf{x}) = f(x_0)+x_1f'(x_0)\varepsilon_1+x_2f'(x_0)\varepsilon_2+x_3f'(x_0)\varepsilon_3+\big(x_{12}f'(x_0)+x_1x_2f''(x_0)\big)\varepsilon_1\varepsilon_2+\big(x_{13}f'(x_0)+x_1x_3f''(x_0)\big)\varepsilon_1\varepsilon_3+\big(x_{23}f'(x_0)$$

A hyperdual number derived type is provided by: hyperdual.

References

[1] Fike, Alonso: **The Development of Hyper-Dual Numbers for Exact Second-Derivative Calculations.** *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition* (2011)
[2] Rehner, P. and Bauer, G.: **Application of Generalized (Hyper-) Dual Numbers in Equation of State Modeling.** Frontiers in Chemical Engineering_ (2021)

5.28 isolines Module Reference

Module for mapping iso-lines.

Functions/Subroutines

- subroutine, public isotherm (t, pmin, pmax, z, n, pa, va, sa, ha, na)

  Map isotherm.
- subroutine, public isobar (p, tmin, tmax, z, n, ta, va, sa, ha, na)

  Map isobar.
- subroutine, public isenthalp (h, pmin, pmax, tmin, tmax, z, n, pa, va, sa, ta, na)

  Map isoenthalp.
- subroutine, public isentrope (s, pmin, pmax, tmin, tmax, z, n, pa, va, ha, ta, na)

  Map isoentrope.

5.28.1 Detailed Description

Module for mapping iso-lines.

5.28.2 Function/Subroutine Documentation

5.28.2.1 isenthalp()

```
subroutine, public isolines::isenthalp (
             real, intent(in) h,
             real, intent(in) pmin,
             real, intent(in) pmax,
             real, intent(in) tmin,
             real, intent(in) tmax,
             real, dimension(nc), intent(in) z,
             integer, intent(in) n,
             real, dimension(n), intent(out) pa,
             real, dimension(n), intent(out) va,
             real, dimension(n), intent(out) sa,
             real, dimension(n), intent(out) ta,
             integer, intent(out) na )
```

Map isoenthalp.

Author

Morten Hammer, 2020-06

Parameters

in	*h*	Specific enthalpy (J/mol)
in	*pmax*	Pressure limits (Pa)
in	*tmax*	Temperature limits (K)
in	*z*	Composition (-)
in	*n*	Number of points
out	*pa*	Temperature (Pa)
out	*va*	Spcecific volume (m3/mol)
out	*sa*	Specific entropy (J/mol/K)
out	*ta*	Temperature (K)
out	*na*	Number of points

5.28.2.2 isentrope()

```
subroutine, public isolines::isentrope (
             real, intent(in) s,
             real, intent(in) pmin,
             real, intent(in) pmax,
             real, intent(in) tmin,
             real, intent(in) tmax,
             real, dimension(nc), intent(in) z,
             integer, intent(in) n,
             real, dimension(n), intent(out) pa,
             real, dimension(n), intent(out) va,
             real, dimension(n), intent(out) ha,
             real, dimension(n), intent(out) ta,
             integer, intent(out) na )
```

Map isoentrope.

Author

Morten Hammer, 2020-06

Parameters

in	*s*	Specific entropy (J/mol/K)
in	*pmax*	Pressure limits (Pa)
in	*tmax*	Temperature limits (K)
in	*z*	Composition (-)
in	*n*	Number of points
out	*pa*	Pressure (Pa)
out	*va*	Spcecific volume (m3/mol)
out	*ha*	Specific enthalpy (J/mol)
out	*ta*	Temperature (K)
out	*na*	Number of points

5.28.2.3 isobar()

```
subroutine, public isolines::isobar (
             real, intent(in) p,
             real, intent(in) tmin,
```

```
            real, intent(in) tmax,
            real, dimension(nc), intent(in) z,
            integer, intent(in) n,
            real, dimension(n), intent(out) ta,
            real, dimension(n), intent(out) va,
            real, dimension(n), intent(out) sa,
            real, dimension(n), intent(out) ha,
            integer, intent(out) na )
```

Map isobar.

Author

Morten Hammer, 2020-06

Parameters

in	*p*	Pressure (Pa)
in	*tmax*	Temperature limits (K)
in	*z*	Composition (-)
in	*n*	Number of points
out	*ta*	Temperature (K)
out	*va*	Spcecific volume (m3/mol)
out	*sa*	Specific entropy (J/mol/K)
out	*ha*	Specifc enthalpy (J/mol)
out	*na*	Number of points

5.28.2.4 isotherm()

```
subroutine, public isolines::isotherm (
            real, intent(in) t,
            real, intent(in) pmin,
            real, intent(in) pmax,
            real, dimension(nc), intent(in) z,
            integer, intent(in) n,
            real, dimension(n), intent(out) pa,
            real, dimension(n), intent(out) va,
            real, dimension(n), intent(out) sa,
            real, dimension(n), intent(out) ha,
            integer, intent(out) na )
```

Map isotherm.

Author

Morten Hammer, 2020-06

Parameters

in	*t*	Temperature (K)
in	*pmax*	Pressure limits (Pa)
in	*z*	Composition (-)
in	*n*	Number of points
out	*pa*	Pressure (Pa)
out	*va*	Spcecific volume (m3/mol)
out	*sa*	Specific entropy (J/mol/K)
out	*ha*	Specifc enthalpy (J/mol)

Parameters

out	*na*	Number of points

5.29 leekesler Module Reference

This module solves the Lee-Kesler EoS. Contains all partial derivative functions needed to do consistency check.

Functions/Subroutines

- subroutine, public mainleekesler (nc, comp, cbeos, t, p, nmoles, phase, z, sdep, hdep, lnphi)

  Main subroutine of the Lee Kesler eos. Takes temperature, pressure, composition and phase as input, and returns comrpessibility, entropy departure, enthalpy departure and fugacity coefficients. Calls upon routine thermProps to calculate these values for simple fluid (0) and reference fluid (1) seperatly, with reduced values for temperature and pressure calculated by the mixing rules, and combines the results according to the Lee Kesler relation:
- subroutine, public lkcalcfug (nc, comp, cbeos, t, p, z, phase, lnfug, dlnfdt, dlnfdp, dlnfdz, v)

  This function calculates the Fugacity coefficient and the derivatives.
- real function lnphim (t, p, tr, pr, vr, moles, b, c, d, e, simporref, dlnphidt, dlnphidp)

  This function calculates reduced deparure Gibbs energy, or the mixture fugacity coefficient. That is; equation 21 in chapter 2 of Michelsen book:
- subroutine, public lkcalcgdep (nc, comp, cbeos, t, p, nmoles, phase, g, dgdt, dgdp)

  This function calculates reduced deparure Gibbs energy.
- subroutine, public lkcalczfac (nc, comp, cbeos, t, p, z, phase, zfac, dzdt, dzdp, dzdz)

  This function calculates the compressibillity factor and the derivatives.
- subroutine, public lkcalcentropy (nc, comp, cbeos, t, p, z, phase, entropy, dsdt, dsdp, dsdz)

  This function calculates the residual entropy and the derivatives.
- subroutine, public lkcalcenthalpy (nc, comp, cbeos, t, p, z, phase, enthalpy, dhdt, dhdp, dhdz)

  This function calculates the residual enthalpy and the derivatives.
- subroutine thermprops (nc, comp, cbeos, tr, pr, vr, nmoles, tcm, vcm, pcm, zcm, wm, moles, z, s, h, lnphi, simporref)

  Subroutine to calculate the thermodynamic properties of either simple fluid or reference fluid. Solutions are compined in main routine.
- subroutine mixrules (nc, comp, cbeos, tcm, vcm, pcm, zcm, wm, nmoles, moles)

  Calculate the critical mixing temperature, volume, pressure, compressibility and acentricity factor, for given composition. Critical temperature and critical volume is used in the calculations, these values are known.
- subroutine trcoeff (tr, b, c, d, e, simporref)

  Calculate the reduced temperature dependent coefficients B,C,D and E, in the expression for Helmholts reduced recidual function F, for given reduced temperature.
- subroutine trcoeffdiff1 (tr, b_tr, c_tr, d_tr, e_tr, simporref)

  Calculate the first order derivatives of the reduced temperature dependent coefficients B,C,D and E, for given reduced temperature.
- subroutine trcoeffdiff2 (tr, b_trtr, c_trtr, d_trtr, e_trtr, simporref)

  Calculate the second order derivatives of the reduced temperature dependent coefficients B,C,D and E, for given reduced temperature.
- subroutine vrnewtraps (tr, pr, usedphase, simporref, vr, correctphase)

  Calculate the reduced volume, vr, by use of the Newton-Rapson numerical method to solve Lee Keslers equation of state, given reduced temperature, reduced pressure and composition.The half step method is used to secure that vr is inside its limit. Calls upon functions fv and fvDiff, acting as the function f and the derivative of f in the numerical method.
- real function fv (pr, tr, vr, b, c, d, e, simporref)

  Calculate the function f to be used in the numerical method to find reduced volume, given reduced pressure, reduced temperature, coefficients B - E and an increasingly more correct reduced volume, for each call.
- real function fvdiff (pr, tr, vr, b, c, d, e, simporref)

Calculate the derivative of function f to be used in the numerical method to find reduced specific volume.

- real function fsolver (moles, vr, b, c, d, e, simporref)

  Calculate the reduced recidual Helmholtz function F, given number of moles, reduced volume and coefficients B - E. This is the main function to be used to find the desired thermodynamic properties.

- real function fdifftr (moles, tr, vr, simporref)

  In the following, functions that give the partial derivatives of F with respect to reduced temperature, reduced specific volume and composition are given.

- real function fdiff2tr (moles, tr, vr, simporref)

  Calculate the second order derivative of Helmholts reduced recidual function, with respect to reduced temperature, for given moles, reduced temperature and reduced specific volume. Volume and compotsition is fixed.

- real function fdiffvr (moles, vr, b, c, d, e, simporref)

  Calculate the first order derivative of Helmholts reduced recidual function, with respect to reduced volume, for given moles, reduced temperature and reduced specific volume. Temperature and compotsition is fixed.

- real function fdiff2vr (moles, vr, b, c, d, e, simporref)

  Calculate the second order derivative of Helmholts reduced recidual function, with respect to reduced volume, for given moles, reduced temperature and reduced specific volume. Temperature and compotsition is fixed.

- real function fdiff3vr (moles, vr, b, c, d, e, simporref)

  Calculate the third order derivative of Helmholtz reduced recidual function, with respect to reduced volume, for given moles, reduced temperature and reduced specific volume. Temperature and compotsition is fixed.

- real function fdiffni (nc, comp, cbeos, tr, vr, tcm, vcm, pcm, zcm, wm, nmoles, moles, i, b, c, d, e, simporref)

  Calculate the first order derivative of Helmholts reduced recidual function, with respect to composition, n(i), for given copmosition, reduced temperature and reduced specific volume. Temperature and volume is fixed.

- real function fdiff2ninj (nc, comp, cbeos, tr, vr, tcm, vcm, pcm, zcm, wm, nmoles, moles, i, j, b, c, d, e, simporref)

  Calculate the second order derivative of Helmholts reduced recidual function, with respect to composition. Temperature and volume is fixed.

- real function fdiffn (moles, vr, b, c, d, e, simporref)

  Help functions to calculate the first and second order derivatives of F with respect to composition.

- real function fdiff2vrn (moles, vr, b, c, d, e, simporref)

  Calculate the cross derivative of F, with respect to reduced volume and total number of moles, for fixed reduced temperature.

- real function fdiff2trn (moles, tr, vr, simporref)

  Calculate the cross derivative of F, with respect to reduced temperature and total number of moles, for fixed reduced volume.

- real function trdiffni (nc, comp, cbeos, tr, tcm, vcm, nmoles, moles, i)

  Calculate the first order derivative of reduced temperature, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.

- real function vrdiffni (nc, comp, cbeos, vr, tcm, vcm, pcm, zcm, wm, nmoles, moles, i)

  Calculate the first order derivative of reduced volume, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.

- real function trdiff2ninj (nc, comp, cbeos, tr, tcm, vcm, nmoles, moles, i, j)

  Calculate the second order derivative of reduced temperature, with respect to composition, for fixed temperature and volume. Help function to be used in finding the second order derivative of F, with respect to composition.

- real function vrdiff2ninj (nc, comp, cbeos, vr, tcm, vcm, pcm, zcm, wm, nmoles, moles, i, j)

  Calculate the second order derivative of reduced volume, with respect to composition, for fixed temperature and volume. Help function to be used in finding the second order derivative of F, with respect to composition.

- real function tcmdiffni (nc, comp, cbeos, tcm, vcm, nmoles, moles, i)

  Calculate the first order derivative of critical mixing temperature, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.

- real function tcmdiff2ninj (nc, comp, cbeos, tcm, vcm, nmoles, moles, i, j)

  Calculate the second order derivative of critical mixing temperature, with respect to composition, for fixed temperature and volume. Help function to be used in finding the second order derivative of F, with respect to composition.

- real function vcmdiffni (nc, comp, nmoles, vcm, moles, i)

Calculate the first order derivative of critical mixing volume, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.

- real function vcmdiff2ninj (nc, comp, nmoles, moles, vcm, i, j)

  Calculate the second order derivative of critical mixing volume, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.
- real function pcmdiffni (nc, comp, cbeos, tcm, vcm, pcm, zcm, wm, nmoles, moles, i)

  Calculate the first order derivative of the pseudo critical mixing pressure, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.
- real function pcmdiff2ninj (nc, comp, cbeos, tcm, vcm, pcm, zcm, wm, nmoles, moles, i, j)

  Calculate the first order derivative of the pseudo critical mixing pressure, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.
- real function zcmdiffni (nc, comp, moles, wm, i)

  Calculate the first order derivative of the pseudo critical mixing compressibility, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.
- real function zcmdiff2ninj (nc, comp, moles, wm, i, j)

  Calculate the second order derivative of the pseudo critical mixing compressibility, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.
- real function wmdiffni (nc, comp, moles, wm, i)

  Calculate the first order derivative of the pseudo critical acentricity factor, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.
- real function wmdiff2ninj (nc, comp, moles, wm, i, j)

  Calculate the second order derivative of the pseudo critical acentricity factor, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.
- real function zdifft (z, t, p, moles, tr, vr, tcm, pcm, b, c, d, e, simporref)

  The derivatives of the compressibility, with respect to temperature, pressure and composition follow.
- real function zdiffp (z, p, moles, tr, vr, tcm, pcm, b, c, d, e, simporref)

  Calculate the derivative of the compressibility, with respect to pressure, for fixed temperature and composition.
- real function zdiffni (nc, comp, cbeos, z, moles, tr, vr, tcm, vcm, pcm, zcm, wm, b, c, d, e, nmoles, i, simporref)

  Calculate the derivative of the compressibility, with respect to composition, for fixed temperature and pressure.
- real function sdifft (t, p, moles, tr, vr, tcm, pcm, b, c, d, e, simporref)

  The derivatives of the entropy, with respect to temperature, pressure and composition follow.
- real function sdiffp (t, p, moles, tr, vr, tcm, pcm, b, c, d, e, simporref)

  Calculate the derivative of entropy, with respect to pressure, for fixed temperature and composition.
- real function sdiffni (nc, comp, cbeos, t, p, moles, tr, vr, tcm, vcm, pcm, zcm, wm, b, c, d, e, nmoles, i, z, simporref)

  Calculate the derivative of entropy, with respect to pressure, for fixed temperature and composition.
- real function hdifft (t, p, moles, tr, vr, tcm, pcm, b, c, d, e, simporref)

  The derivatives of the enthalpy, with respect to temperature, pressure and composition follow.
- real function hdiffp (t, p, moles, tr, vr, tcm, pcm, b, c, d, e, simporref)

  Calculate the derivative of enthalpy, with respect to pressure, for fixed temperature and composition.
- real function hdiffni (nc, comp, cbeos, t, p, moles, tr, vr, tcm, vcm, pcm, zcm, wm, b, c, d, e, nmoles, i, simporref)

  Calculate the derivative of enthalpy, with respect to composition, for fixed temperature and pressure.
- real function lnphidifft (nc, comp, cbeos, t, p, moles, tr, vr, tcm, vcm, pcm, zcm, wm, b, c, d, e, nmoles, i, simporref)

  The derivatives of the logarithm of the fugacity coefficient, with respect to temperature, pressure and composition follow.
- real function lnphidiffp (nc, comp, cbeos, t, p, moles, tr, vr, tcm, vcm, pcm, zcm, wm, b, c, d, e, nmoles, i, simporref)

  Calculate the derivative of the logarithmic fugacity coefficients, with respect to pressure, for fixed temperature and composition.
- real function lnphidiffnj (nc, comp, cbeos, moles, tr, vr, tcm, vcm, pcm, zcm, wm, b, c, d, e, nmoles, i, j, simporref)

Calculate the derivative of the logarithmic fugacity coefficients, with respect to composition, for fixed temperature and pressure.

- real function fdiff2tni (nc, comp, cbeos, moles, tr, vr, tcm, vcm, pcm, zcm, wm, nmoles, i, simporref)

  Help functions to calculate the derivatives of the entropy, enthalpy and logarithm of the fugacity coeficients.
- real function fdiff2trvr (moles, tr, vr, simporref)

  Calculate the derivative of Helmholts reduced recidual function, with respect to reduced temperature and reduced specific volume. Composition is fixed.
- real function fdiff2vni (nc, comp, cbeos, moles, tr, vr, tcm, vcm, pcm, zcm, wm, nmoles, b, c, d, e, i, simporref)

  Calculate the derivative of Helmholts reduced recidual function, with respect to composition and volume. Temperature is fixed.
- real function vdiffni (nc, comp, cbeos, moles, tr, vr, tcm, vcm, pcm, zcm, wm, b, c, d, e, nmoles, i, simporref)

  Calculate the derivative of the volume, with respect to composition, For fixed temperature and pressure. Help function that simplifies notation.
- real function vdifft (t, p, moles, tr, vr, tcm, pcm, b, c, d, e, simporref)

  Calculate the derivative of the volume, with respect to temperature, For fixed pressire and composition. Help function that simplifies notation.
- real function pdifft (t, p, moles, tr, vr, tcm, pcm, simporref)

  Pressure P = P(T,V,n) Derivatives of P with respect to temperature, volume and composition are used when finding several of the derivatives of the thermodynamic properties of interest. The functions for these derivatives follow.
- real function pdiffv (moles, tr, vr, tcm, pcm, b, c, d, e, simporref)

  Calculate the derivative of the pressure, with respect to volume, for fixed temperature and composition.
- real function pdiffni (nc, comp, cbeos, moles, tr, vr, tcm, vcm, pcm, zcm, wm, b, c, d, e, nmoles, i, simporref)

  Calculate the derivative of the pressure, with respect to composition, for fixed temperature and volume.
- subroutine vrinitial (pr, tr, usedphase, simporref, vrinit, vrmin, vrmax)

  The following subroutine is a routine that helps solving the non-linear equation of state (i.e. finding the reduced volume) by calculationg an appropriate initial value to be used in the numerical method. A bound for the allowed value of the reduced specific volume is also found.
- subroutine zprtshape (t, tr, b, c, d, e, simporref)

  Optional subroutines and functions. These arenot called upon automatically, hence the thermProps routine should be modified to call upon them, if this output is desired.
- subroutine **fvshape** (t, p, tr, pr, b, c, d, e, simporref)
- real function prsolver (tr, vr, b, c, d, e, simporref)

  Calculate the reduced pressure, Pr, by directly solving the Lee Keslers equation of state, given reduced temperature, reduced volume and composition.
- subroutine znewtraps (tr, pr, phase, simporref, vr, zinit, zmax, zmin)

  Calculate the reduced volume, vr, by use of the Newton-Rapson numerical method to solve Lee Keslers equation of state, given reduced temperature, reduced pressure and composition.The half step method is used to secure that vr is inside its limit. Calls upon functions fz, fzDiff and fzDiff2 acting as the function f, derivative of f and second derivative in the numerical method.
- subroutine zinitial (tr, pr, phase, simporref, z, zmin, zmax, solved, hasphase)

  Find phase minima/maxima of the Lee Keslers equation of state. If f(z) = 0 have a solution an interval for the solution is returned. If the search don't cross f(z) = 0, the minima/maxima is returned. Calls upon functions fz, fzDiff and fzDiff2 acting as the function f, derivative of f and second derivative in the numerical method.
- real function fz (pr, tr, z, b, c, d, e, simporref)

  Calculate the function f to be used in the numerical method to find compressibility, given reduced pressure, reduced temperature, coefficients B - E and an increasingly more correct compressibility, for each call.
- real function fzdiff (pr, tr, z, b, c, d, e, simporref)

  Calculate the derivative of function f to be used in the numerical method to find compressibillity.
- real function fzdiff2 (pr, tr, z, b, c, d, e, simporref)

  Calculate the second derivative of function f to be used in the numerical method to find compressibility.
- subroutine fzwithdiff (pr, tr, z, b, c, d, e, simporref, fz, fzd, fzd2)

  Calculate the function f to be used in the numerical method to find compressibillity. Differentials is also calculated, see fz, fzDiff and fzDiff2 for details.
- subroutine, public testdiffleekesler (nc, comp, cbeos, tin, pin, z, phase)

Test differentials of the Lee Kesler eos. Takes temperature, pressure, composition and phase as input, and test compressibility, entropy departure, enthalpy departure and fugacity coefficients differentials.

- subroutine calcreducedvolume (tr, pr, phase, vrsimp, vrref)

  Calculate the reduced volume, vr, by use of the Newton-Rapson numerical method to solve Lee Keslers equation of state, given reduced temperature, reduced pressure and composition.The half step method is used to secure that vr is inside its limit. Calls upon functions fz, fzDiff and fzDiff2 acting as the function f, derivative of f and second derivative in the numerical method.
- subroutine setmaxminz (tr, pr, phase, z, zmax, zmin)

  Set limits for compressibillity factor, and initial value based on phase flag.
- subroutine fixedtrplot (vrlow, vrhigh, tr, outfile)

  Dump data to file for plotting.
- real function pred (tr, vr, simporref)

  Calculate the reduced pressure given reduzed temperature and reduced volume:

Variables

- real, parameter **eta** = 0.25
- real, parameter **wref** = 0.3978
- integer, parameter **rootselection** = 3

  1: No phase root analysis 2: Return minima if no root exsist 3: If the Simple/Reference phase don't have a root, but have a minima switch to other phase
- real, parameter **vrminimum** = 0.05
- real, parameter **vrmaximum** = 1E10
- real, dimension(2), parameter **b1** = (/ 0.1181193, 0.2026579 /)

  The constants b1, b2, b3, b4, c1, c2, c3, c4, d1, d2, beta and gamma have two distinct values each. Whether the first or second stored value for the constants are used, is determined by whether the calculations are done for a simple fluid, or a reference fluid. 1=simple fluid, 2=reference fluid.
- real, dimension(2), parameter **b2** = (/ 0.265728, 0.331511 /)
- real, dimension(2), parameter **b3** = (/ 0.154790, 0.027655 /)
- real, dimension(2), parameter **b4** = (/ 0.030323, 0.203488 /)
- real, dimension(2), parameter **c1** = (/ 0.0236744, 0.0313385 /)
- real, dimension(2), parameter **c2** = (/ 0.0186984, 0.0503618 /)
- real, dimension(2), parameter **c3** = (/ 0.0, 0.016901 /)
- real, dimension(2), parameter **c4** = (/ 0.042724, 0.041577/)
- real, dimension(2), parameter **d1** = (/ 0.0000155488, 0.000048736 /)
- real, dimension(2), parameter **d2** = (/ 0.0000623689, 0.00000740336/)
- real, dimension(2), parameter **beta** = (/ 0.65392, 1.226 /)
- real, dimension(2), parameter **gamma** = (/ 0.060167, 0.03754 /)

5.29.1 Detailed Description

This module solves the Lee-Kesler EoS. Contains all partial derivative functions needed to do consistency check.

Author

JA, June - July 2013

5.29.2 Function/Subroutine Documentation

5.29.2.1 calcreducedvolume()

```
subroutine leekesler::calcreducedvolume (
             real, intent(in) tr,
             real, intent(in) pr,
             integer, intent(in) phase,
             real, intent(out) vrsimp,
             real, intent(out) vrref )
```

Calculate the reduced volume, vr, by use of the Newton-Rapson numerical method to solve Lee Keslers equation of state, given reduced temperature, reduced pressure and composition.The half step method is used to secure that vr is inside its limit. Calls upon functions fz, fzDiff and fzDiff2 acting as the function f, derivative of f and second derivative in the numerical method.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}\left(1 + \frac{f(x_n)f''(x_n)}{f'(x_n)^2}\right)$$

A Taylor expansion of f is used, and the root giving the smallest change in x is used. To simplify the step, the following series approximation is used,

$$1 - \sqrt{1-\alpha} = \frac{\alpha}{2} + \frac{\alpha^2}{8} + O(\alpha^3).$$

Author

MH, 2013-09

5.29.2.2 fdiff2ninj()

```
real function leekesler::fdiff2ninj (
             integer, intent(in) nc,
             type (gendata_pointer), dimension(nc) comp,
             class(cb_eos), intent(in) cbeos,
             real, intent(in) tr,
             real, intent(in) vr,
             real, intent(in) tcm,
             real, intent(in) vcm,
             real, intent(in) pcm,
             real, intent(in) zcm,
             real, intent(in) wm,
             real, dimension(nc), intent(in) nmoles,
             real, intent(in) moles,
             integer, intent(in) i,
             integer, intent(in) j,
             real, intent(in) b,
             real, intent(in) c,
             real, intent(in) d,
             real, intent(in) e,
             integer, intent(in) simporref )
```

Calculate the second order derivative of Helmholts reduced recidual function, with respect to composition. Temperature and volume is fixed.

$$\begin{aligned} F_{ij} = \left(\frac{\partial^2 F}{\partial n_i \partial n_j}\right)_{T,V} &= F_{NN} + F_{NX}X_j + F_{NY}Y_j \\ &+ \left(F_{NX} + F_{XY}Y_j + F_{XX}X_j\right)X_i + F_X X_{ij} \\ &+ \left(F_{NY} + F_{YY}Y_j + F_{XY}X_j\right)Y_i + F_Y Y_{ij} \end{aligned}$$

With: X = T_r Y = v_r N = n

Author

JA, 2013-06

5.29.2.3 fdiff2tni()

```
real function leekesler::fdiff2tni (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc) comp,
            class(cb_eos), intent(in) cbeos,
            real, intent(in) moles,
            real, intent(in) tr,
            real, intent(in) vr,
            real, intent(in) tcm,
            real, intent(in) vcm,
            real, intent(in) pcm,
            real, intent(in) zcm,
            real, intent(in) wm,
            real, dimension(nc), intent(in) nmoles,
            integer, intent(in) i,
            integer, intent(in) simporref )
```

Help functions to calculate the derivatives of the entropy, enthalpy and logarithm of the fugacity coeficients. Calculate the derivative of Helmholts reduced recidual function, with respect to composition and temperature. Volume is fixed.

$$F_{iT} = \left(\frac{\partial^2 F}{\partial T \partial n_i}\right)_V = F_{NT} + F_{NX}X_T + F_{NY}Y_T + (F_{XT} + F_{XX}X_T + F_{XY}Y_T)X_i + F_X X_{iT} + (F_{YT} + F_{YX}X_T + F_{YY}Y_T)Y_i + F_Y Y_{iT}$$

Author

JA, 2013-07

5.29.2.4 fdiff2tr()

```
real function leekesler::fdiff2tr (
            real, intent(in) moles,
            real, intent(in) tr,
            real, intent(in) vr,
            integer, intent(in) simporref )
```

Calculate the second order derivative of Helmholts reduced recidual function, with respect to reduced temperature, for given moles, reduced temperature and reduced specific volume. Volume and compotsition is fixed.

$$\left(\frac{\partial^2 F}{\partial T_r^2}\right)_{v_r,n} = n\left[\frac{B_{TT_r}}{v_r} + \frac{C_{TT_r}}{2v_r^2} + \frac{D_{TT_r}}{5v_r^5} + \frac{E_{TT_r}}{2\gamma}(\beta+1) - E_{TT_r}\exp\left(-\frac{\gamma}{v_r^2}\right)\left(\frac{1}{2\gamma}(\beta+1) + \frac{1}{2v_r^2}\right)\right]$$

Author

JA, 2013-06

5.29.2.5 fdiff2trn()

```
real function leekesler::fdiff2trn (
            real, intent(in) moles,
```

```
            real, intent(in) tr,
            real, intent(in) vr,
            integer, intent(in) simporref )
```

Calculate the cross derivative of F, with respect to reduced temperature and total number of moles, for fixed reduced volume.

$$F_{nX} = \left(\frac{\partial^2 F}{\partial n \partial T_r}\right)_{v_r} = \frac{1}{n}\left(\frac{\partial F}{\partial T_r}\right)_{v_r,n}$$

Author

JA, 2013-07

5.29.2.6 fdiff2trvr()

```
real function leekesler::fdiff2trvr (
            real, intent(in) moles,
            real, intent(in) tr,
            real, intent(in) vr,
            integer, intent(in) simporref )
```

Calculate the derivative of Helmholts reduced recidual function, with respect to reduced temperature and reduced specific volume. Composition is fixed.

$$F_{XY} = \left(\frac{\partial^2 F}{\partial T_r \partial v_r}\right)_n = -n\left[\frac{B_{T_r}}{v_r^2} + \frac{C_{T_r}}{v_r^3} + \frac{D_{T_r}}{v_r^6} + \frac{E_{T_r}}{v_r^3}\left(\beta + \frac{\gamma}{v_r^2}\right)\exp\left(-\frac{\gamma}{v_r^2}\right)\right]$$

Author

JA, 2013-07

5.29.2.7 fdiff2vni()

```
real function leekesler::fdiff2vni (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc) comp,
            class(cb_eos), intent(in) cbeos,
            real, intent(in) moles,
            real, intent(in) tr,
            real, intent(in) vr,
            real, intent(in) tcm,
            real, intent(in) vcm,
            real, intent(in) pcm,
            real, intent(in) zcm,
            real, intent(in) wm,
            real, dimension(nc), intent(in) nmoles,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            integer, intent(in) i,
            integer, intent(in) simporref )
```

Calculate the derivative of Helmholts reduced recidual function, with respect to composition and volume. Temperature is fixed.

$$F_{iV} = \left(\frac{\partial^2 F}{\partial V \partial n_i}\right)_T = F_{NV} + F_{NX}X_V + F_{NY}Y_V + (F_{XV} + F_{XX}X_V + F_{XY}Y_V)X_i + F_X X_{iV} + (F_{YV} + F_{YX}X_V + F_{YY}Y_V)Y_i + F_Y Y_{iV}$$

Author

JA, 2013-07

5.29.2.8 fdiff2vr()

```
real function leekesler::fdiff2vr (
             real, intent(in) moles,
             real, intent(in) vr,
             real, intent(in) b,
             real, intent(in) c,
             real, intent(in) d,
             real, intent(in) e,
             integer, intent(in) simporref )
```

Calculate the second order derivative of Helmholts reduced recidual function, with respect to reduced volume, for given moles, reduced temperature and reduced specific volume. Temperature and compotsition is fixed.

$$\left(\frac{\partial^2 F}{\partial v_r^2}\right)_{T_r,n} = n\left[\frac{2B}{v_r^3} + \frac{3C}{v_r^4} + \frac{6D}{v_r^7} + \frac{E}{v_r^4}\left(3\beta + \frac{\gamma(5-2\beta)}{v_r^2} - \frac{2\gamma^2}{v_r^4}\right)\exp\left(-\frac{\gamma}{v_r^2}\right)\right]$$

Author

JA, 2013-06

5.29.2.9 fdiff2vrn()

```
real function leekesler::fdiff2vrn (
             real, intent(in) moles,
             real, intent(in) vr,
             real, intent(in) b,
             real, intent(in) c,
             real, intent(in) d,
             real, intent(in) e,
             integer, intent(in) simporref )
```

Calculate the cross derivative of F, with respect to reduced volume and total number of moles, for fixed reduced temperature.

$$F_{NY} = \left(\frac{\partial^2 F}{\partial n \partial v_r}\right)_{T_r} = \left(\frac{\partial}{\partial n}\left(\frac{\partial F}{\partial v_r}\right)_{T_r,n}\right)_{T_r,v_r} = \frac{1}{n}\left(\frac{\partial F}{\partial v_r}\right)_{T_r,n}$$

Author

JA, 2013-07

5.29.2.10 fdiff3vr()

```
real function leekesler::fdiff3vr (
            real, intent(in) moles,
            real, intent(in) vr,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            integer, intent(in) simporref )
```

Calculate the third order derivative of Helmholtz reduced recidual function, with respect to reduced volume, for given moles, reduced temperature and reduced specific volume. Temperature and compotsition is fixed.

$$\left(\frac{\partial^3 F}{\partial v_r^3}\right)_{T_r,n} = n\left[-\frac{6B}{v_r^4}-\frac{12C}{v_r^5}-\frac{42D}{v_r^8} - \frac{E}{v_r^5}\left(12\beta+\frac{(30-18\beta)\gamma}{v_r^2}+\frac{(4\beta-26)\,\gamma^2}{v_r^4}+\frac{4\gamma^3}{v_r^6}\right)\exp\left(-\frac{\gamma}{v_r^2}\right)\right]$$

Author

MH, 2013-09

5.29.2.11 fdiffn()

```
real function leekesler::fdiffn (
            real, intent(in) moles,
            real, intent(in) vr,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            integer, intent(in) simporref )
```

Help functions to calculate the first and second order derivatives of F with respect to composition.
Calculate the derivative of Helmholts reduced recidual function, with respect to total number of moles, for given moles, reduced temperature and reduced specific volume. Reduced temperature and reduced volume are fixed in this derivative.

$$F_N = \left(\frac{\partial F}{\partial n}\right)_{T_r,v_r} = \frac{F}{n}$$

Author

JA, 2013-06

5.29.2.12 fdiffni()

```
real function leekesler::fdiffni (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc) comp,
            class(cb_eos), intent(in) cbeos,
            real, intent(in) tr,
            real, intent(in) vr,
            real, intent(in) tcm,
            real, intent(in) vcm,
            real, intent(in) pcm,
```

```
            real, intent(in) zcm,
            real, intent(in) wm,
            real, dimension(nc), intent(in) nmoles,
            real, intent(in) moles,
            integer, intent(in) i,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            integer, intent(in) simporref )
```

Calculate the first order derivative of Helmholts reduced recidual function, with respect to composition, n(i), for given copmosition, reduced temperature and reduced specific volume. Temperature and volume is fixed.

$$F_i = \left(\frac{\partial F}{\partial n_i}\right)_{T,V} = F_N + F_X X_i + F_Y Y_i$$

Author

JA, 2013-06

5.29.2.13 fdifftr()

```
real function leekesler::fdifftr (
            real, intent(in) moles,
            real, intent(in) tr,
            real, intent(in) vr,
            integer, intent(in) simporref )
```

In the following, functions that give the partial derivatives of F with respect to reduced temperature, reduced specific volume and composition are given.
Calculate the first order derivative of Helmholts reduced recidual function, with respect to reduced temperature, for given moles, reduced temperature and reduced specific volume. Volume and compotsition is fixed.

$$\left(\frac{\partial F}{\partial T_r}\right)_{v_r,n} = n\left[\frac{B_{T_r}}{v_r} + \frac{C_{T_r}}{2v_r^2} + \frac{D_{T_r}}{5v_r^5} + \frac{E_{T_r}}{2\gamma}(\beta+1) - E_{T_r}\exp\left(-\frac{\gamma}{v_r^2}\right)\left(\frac{1}{2\gamma}(\beta+1) + \frac{1}{2v_r^2}\right)\right]$$

Author

JA, 2013-06

5.29.2.14 fdiffvr()

```
real function leekesler::fdiffvr (
            real, intent(in) moles,
            real, intent(in) vr,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            integer, intent(in) simporref )
```

Calculate the first order derivative of Helmholts reduced recidual function, with respect to reduced volume, for given moles, reduced temperature and reduced specific volume. Temperature and compotsition is fixed.

$$\left(\frac{\partial F}{\partial v_r}\right)_{T_r,n} = n\left[-\frac{B}{v_r^2} - \frac{C}{v_r^3} - \frac{D}{v_r^6} - \frac{E}{v_r^3}\left(\beta + \frac{\gamma}{v_r^2}\right)\exp\left(-\frac{\gamma}{v_r^2}\right)\right]$$

Author

JA, 2013-06

5.29.2.15 fixedtrplot()

```
subroutine leekesler::fixedtrplot (
             real, intent(in) vrlow,
             real, intent(in) vrhigh,
             real, intent(in) tr,
             character(len=*), intent(in), optional outfile )
```

Dump data to file for plotting.

Author

MH, 2013-09

5.29.2.16 fsolver()

```
real function leekesler::fsolver (
             real, intent(in) moles,
             real, intent(in) vr,
             real, intent(in) b,
             real, intent(in) c,
             real, intent(in) d,
             real, intent(in) e,
             integer, intent(in) simporref )
```

Calculate the reduced recidual Helmholtz function F, given number of moles, reduced volume and coefficients B - E. This is the main function to be used to find the desired thermodynamic properties.

$$F(T,V,\mathbf{n}) = n\left[\frac{B}{v_r} + \frac{C}{2v_r^2} + \frac{D}{5v_r^5} + \frac{E}{2\gamma}(\beta+1) - E\exp\left(-\frac{\gamma}{v_r^2}\right)\left(\frac{1}{2\gamma}(\beta+1) + \frac{1}{2v_r^2}\right)\right]$$

Author

JA, 2013-07

5.29.2.17 fv()

```
real function leekesler::fv (
             real, intent(in) pr,
             real, intent(in) tr,
             real, intent(in) vr,
             real, intent(in) b,
             real, intent(in) c,
             real, intent(in) d,
             real, intent(in) e,
             integer, intent(in) simporref )
```

Calculate the function f to be used in the numerical method to find reduced volume, given reduced pressure, reduced temperature, coefficients B - E and an increasingly more correct reduced volume, for each call.

$$f = v_r - \frac{T_r}{P_r}\left[1 + \frac{B}{v_r} + \frac{C}{v_r^2} + \frac{D}{v_r^5} + \frac{E}{v_r^2}\left(\beta + \frac{\gamma}{v_r^2}\right)\exp\left(-\frac{\gamma}{v_r^2}\right)\right] = 0$$

Author

JA, 2013-06

5.29.2.18 fvdiff()

```
real function leekesler::fvdiff (
             real, intent(in) pr,
             real, intent(in) tr,
             real, intent(in) vr,
             real, intent(in) b,
             real, intent(in) c,
             real, intent(in) d,
             real, intent(in) e,
             integer, intent(in) simporref )
```

Calculate the derivative of function f to be used in the numerical method to find reduced specific volume.

$$\left(\frac{\partial f}{\partial v_r}\right)_{T_r,P_r,\mathbf{n}} = 1 + \frac{Pr}{Tr}\left[\frac{B}{v_r} + \frac{2C}{v_r^3} + \frac{5D}{v_r^6} - \frac{E}{v_r^3}\left(-2\beta + \frac{\gamma(\beta-4)}{v_r^2} + \frac{2\gamma^2}{v_r^4}\right)\exp\left(-\frac{\gamma}{v_r^2}\right)\right]$$

Author

JA, 2013-06

5.29.2.19 fz()

```
real function leekesler::fz (
             real, intent(in) pr,
             real, intent(in) tr,
             real, intent(in) z,
             real, intent(in) b,
             real, intent(in) c,
             real, intent(in) d,
             real, intent(in) e,
             integer, intent(in) simporref )
```

Calculate the function f to be used in the numerical method to find compressibility, given reduced pressure, reduced temperature, coefficients B - E and an increasingly more correct compressibility, for each call.

$$f = \frac{v_r P_r}{T_r} - \left[1 - \frac{v_r}{n}\left(\frac{\partial F}{\partial v_r}\right)_{T_r,n}\right] = 0$$

Author

MH, 2013-09

5.29.2.20 fzdiff()

```
real function leekesler::fzdiff (
             real, intent(in) pr,
             real, intent(in) tr,
             real, intent(in) z,
             real, intent(in) b,
             real, intent(in) c,
             real, intent(in) d,
             real, intent(in) e,
             integer, intent(in) simporref )
```

Calculate the derivative of function f to be used in the numerical method to find compressibillity.

$$\left(\frac{\partial f}{\partial z}\right)_{T_r,P_r,\mathbf{n}} = 1 + \frac{Tr}{Prn}\left[\left(\frac{\partial F}{\partial v_r}\right)_{T_r,n} + v_r\left(\frac{\partial^2 F}{\partial v_r^2}\right)_{T_r,n}\right]$$

Author

MH, 2013-09

5.29.2.21 fzdiff2()

```
real function leekesler::fzdiff2 (
            real, intent(in) pr,
            real, intent(in) tr,
            real, intent(in) z,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            integer, intent(in) simporref )
```

Calculate the second derivative of function f to be used in the numerical method to find compressibility.

$$\left(\frac{\partial^2 f}{\partial z^2}\right)_{T_r,P_r,\mathbf{n}} = \frac{Tr^2}{Pr^2 n}\left[2\left(\frac{\partial^2 F}{\partial v_r^2}\right)_{T_r,n} + v_r\left(\frac{\partial^3 F}{\partial v_r^3}\right)_{T_r,n}\right]$$

Author

MH, 2013-09

5.29.2.22 fzwithdiff()

```
subroutine leekesler::fzwithdiff (
            real, intent(in) pr,
            real, intent(in) tr,
            real, intent(in) z,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            integer, intent(in) simporref,
            real, intent(out) fz,
            real, intent(out) fzd,
            real, intent(out) fzd2 )
```

Calculate the function f to be used in the numerical method to find compressibillity. Differentials is also calculated, see fz, fzDiff and fzDiff2 for details.

Author

MH, 2013-09

5.29.2.23 hdiffni()

```
real function leekesler::hdiffni (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc) comp,
            class(cb_eos), intent(in) cbeos,
            real, intent(in) t,
            real, intent(in) p,
            real, intent(in) moles,
            real, intent(in) tr,
            real, intent(in) vr,
            real, intent(in) tcm,
            real, intent(in) vcm,
```

```
            real, intent(in) pcm,
            real, intent(in) zcm,
            real, intent(in) wm,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            real, dimension(nc), intent(in) nmoles,
            integer, intent(in) i,
            integer, intent(in) simporref )
```

Calculate the derivative of enthalpy, with respect to composition, for fixed temperature and pressure.

$$\left(\frac{\partial H^R}{\partial n_i}\right)_{T,P} = \bar{V}_i T \left(\frac{\partial P}{\partial T}\right)_{V,\mathbf{n}} - RT^2 \left(\frac{\partial^2 F}{\partial T \partial n_i}\right)_V - RT$$

Author

JA, 2013-07

5.29.2.24 hdiffp()

```
real function leekesler::hdiffp (
            real, intent(in) t,
            real, intent(in) p,
            real, intent(in) moles,
            real, intent(in) tr,
            real, intent(in) vr,
            real, intent(in) tcm,
            real, intent(in) pcm,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            integer, intent(in) simporref )
```

Calculate the derivative of enthalpy, with respect to pressure, for fixed temperature and composition.

$$\left(\frac{\partial H^R}{\partial P}\right)_{T,\mathbf{n}} = V - T\bar{V}_T$$

Author

JA, 2013-07

5.29.2.25 hdifft()

```
real function leekesler::hdifft (
            real, intent(in) t,
            real, intent(in) p,
            real, intent(in) moles,
            real, intent(in) tr,
            real, intent(in) vr,
            real, intent(in) tcm,
            real, intent(in) pcm,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            integer, intent(in) simporref )
```

The derivatives of the enthalpy, with respect to temperature, pressure and composition follow.
Calculate the derivative of enthalpy, with respect to temperature, for fixed pressure and composition.

$$\left(\frac{\partial H^R}{\partial T}\right)_{P,\mathbf{n}} = \bar{V}_T T \left(\frac{\partial P}{\partial T}\right)_{V,\mathbf{n}} - RT\left[2\left(\frac{\partial F}{\partial T}\right)_{V,\mathbf{n}} + T\left(\frac{\partial^2 F}{\partial T^2}\right)_{V,\mathbf{n}} + \frac{n}{T}\right]$$

Author

JA, 2013-07

5.29.2.26 lkcalcenthalpy()

```
subroutine, public leekesler::lkcalcenthalpy (
             integer, intent(in) nc,
             type (gendata_pointer), dimension(nc) comp,
             class(cb_eos), intent(in) cbeos,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             integer, intent(in) phase,
             real, intent(out) enthalpy,
             real, intent(out), optional dhdt,
             real, intent(out), optional dhdp,
             real, dimension(nc), intent(out), optional dhdz )
```

This function calculates the residual enthalpy and the derivatives.

Author

Morten H.

Parameters

in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*z*	The overall mole fraction [-]
in	*phase*	Phase, 1=liquid, 2=vapour
out	*enthalpy*	The residual enthalpy [J/mol]
out	*dhdt*	Temperature derivative [J/mol/K]
out	*dhdp*	Pressure derivative [J/mol/Pa]
out	*dhdz*	Composition derivative [J/mol]

5.29.2.27 lkcalcentropy()

```
subroutine, public leekesler::lkcalcentropy (
             integer, intent(in) nc,
             type (gendata_pointer), dimension(nc) comp,
             class(cb_eos), intent(in) cbeos,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             integer, intent(in) phase,
             real, intent(out) entropy,
             real, intent(out), optional dsdt,
             real, intent(out), optional dsdp,
             real, dimension(nc), intent(out), optional dsdz )
```

This function calculates the residual entropy and the derivatives.

Author

Morten H.

Parameters

in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*z*	The overall mole fraction [-]
in	*phase*	Phase, 1=liquid, 2=vapour
out	*entropy*	The residual entropy [J/mol/K]
out	*dsdt*	Temperature derivative [J/mol/K^2]
out	*dsdp*	Pressure derivative [J/mol/K/Pa]
out	*dsdz*	Composition derivative [J/mol/K]

5.29.2.28 lkcalcfug()

```
subroutine, public leekesler::lkcalcfug (
             integer, intent(in) nc,
             type (gendata_pointer), dimension(nc) comp,
             class(cb_eos), intent(in) cbeos,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             integer, intent(in) phase,
             real, dimension(nc), intent(out) lnfug,
             real, dimension(nc), intent(out), optional dlnfdt,
             real, dimension(nc), intent(out), optional dlnfdp,
             real, dimension(nc,nc), intent(out), optional dlnfdz,
             real, intent(out), optional v )
```

This function calculates the Fugacity coefficient and the derivatives.

Author

Morten H.

Parameters

in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*z*	The overall mole fraction [-]
in	*phase*	Phase, 1=liquid, 2=vapour
out	*lnfug*	The fugacity coefficients [-]
out	*dlnfdt*	Temperature derivative [1/K] (dln(f)/dT)
out	*dlnfdp*	Pressure derivative [1/Pa] (dln(f)/dP)
out	*dlnfdz*	Composition derivative [1/kmole] (dln(f)/dNi)
out	*v*	Specific volume [mol/m3]

5.29.2.29 lkcalcgdep()

```
subroutine, public leekesler::lkcalcgdep (
```

```
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc) comp,
            class(cb_eos), intent(in) cbeos,
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) nmoles,
            integer, intent(in) phase,
            real, intent(out) g,
            real, intent(out), optional dgdt,
            real, intent(out), optional dgdp )
```

This function calculates reduced deparure Gibbs energy.

Author

Morten H.

Parameters

in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*nmoles*	Number of moles per component [mol]
in	*phase*	Phase integer
out	*g*	The departure gibbs energy [J/mol]

5.29.2.30 lkcalczfac()

```
subroutine, public leekesler::lkcalczfac (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc) comp,
            class(cb_eos), intent(in) cbeos,
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) z,
            integer, intent(in) phase,
            real, intent(out) zfac,
            real, intent(out), optional dzdt,
            real, intent(out), optional dzdp,
            real, dimension(nc), intent(out), optional dzdz )
```

This function calculates the compressibillity factor and the derivatives.

Author

Morten H.

Parameters

in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*z*	The overall mole fraction [-]
in	*phase*	Phase, 1=liquid, 2=vapour
out	*zfac*	The compressibilltiy factor [-]
out	*dzdt*	Temperature derivative [1/K]
out	*dzdp*	Pressure derivative [1/Pa]
out	*dzdz*	Composition derivative [-]

5.29.2.31 lnphidiffnj()

```
real function leekesler::lnphidiffnj (
              integer, intent(in) nc,
              type (gendata_pointer), dimension(nc) comp,
              class(cb_eos), intent(in) cbeos,
              real, intent(in) moles,
              real, intent(in) tr,
              real, intent(in) vr,
              real, intent(in) tcm,
              real, intent(in) vcm,
              real, intent(in) pcm,
              real, intent(in) zcm,
              real, intent(in) wm,
              real, intent(in) b,
              real, intent(in) c,
              real, intent(in) d,
              real, intent(in) e,
              real, dimension(nc), intent(in) nmoles,
              integer, intent(in) i,
              integer, intent(in) j,
              integer, intent(in) simporref )
```

Calculate the derivative of the logarithmic fugacity coefficients, with respect to composition, for fixed temperature and pressure.

$$\left(\frac{\partial \ln \phi_i}{\partial n_i}\right)_{T,P} = \left(\frac{\partial^2 F}{\partial n_j \partial n_i}\right)_{T,P} + \frac{1}{n} + \frac{\left(\frac{\partial P}{\partial V}\right)_{T,\mathbf{n}}}{RT} \bar{V}_j \bar{V}_i$$

Author

JA, 2013-07

5.29.2.32 lnphidiffp()

```
real function leekesler::lnphidiffp (
              integer, intent(in) nc,
              type (gendata_pointer), dimension(nc) comp,
              class(cb_eos), intent(in) cbeos,
              real, intent(in) t,
              real, intent(in) p,
              real, intent(in) moles,
              real, intent(in) tr,
              real, intent(in) vr,
              real, intent(in) tcm,
              real, intent(in) vcm,
              real, intent(in) pcm,
              real, intent(in) zcm,
              real, intent(in) wm,
              real, intent(in) b,
              real, intent(in) c,
              real, intent(in) d,
              real, intent(in) e,
              real, dimension(nc), intent(in) nmoles,
              integer, intent(in) i,
              integer, intent(in) simporref )
```

Calculate the derivative of the logarithmic fugacity coefficients, with respect to pressure, for fixed temperature and composition.

$$\left(\frac{\partial \ln \phi_i}{\partial P}\right)_{T,\mathbf{n}} = \frac{\bar{V}_i}{RT} - \frac{1}{P}$$

Author

JA, 2013-07

5.29.2.33 lnphidifft()

```
real function leekesler::lnphidifft (
             integer, intent(in) nc,
             type (gendata_pointer), dimension(nc) comp,
             class(cb_eos), intent(in) cbeos,
             real, intent(in) t,
             real, intent(in) p,
             real, intent(in) moles,
             real, intent(in) tr,
             real, intent(in) vr,
             real, intent(in) tcm,
             real, intent(in) vcm,
             real, intent(in) pcm,
             real, intent(in) zcm,
             real, intent(in) wm,
             real, intent(in) b,
             real, intent(in) c,
             real, intent(in) d,
             real, intent(in) e,
             real, dimension(nc), intent(in) nmoles,
             integer, intent(in) i,
             integer, intent(in) simporref )
```

The derivatives of the logarithm of the fugacity coefficient, with respect to temperature, pressure and composition follow.
Calculate the derivative of the logarithmic fugacity coefficients, with respect to temperature, for fixed pressure and composition.

$$\left(\frac{\partial \ln \phi_i}{\partial T}\right)_{P,\mathbf{n}} = \left(\frac{\partial^2 F}{\partial T \partial n_i}\right)_V + \frac{1}{T} - \frac{\bar{V}_i}{RT}\left(\frac{\partial P}{\partial T}\right)_{V,\mathbf{n}}$$

Author

JA, 2013-07

5.29.2.34 lnphim()

```
real function leekesler::lnphim (
             real, intent(in) t,
             real, intent(in) p,
             real, intent(in) tr,
             real, intent(in) pr,
             real, intent(in) vr,
             real, intent(in) moles,
             real, intent(in) b,
             real, intent(in) c,
             real, intent(in) d,
             real, intent(in) e,
             integer, intent(in) simporref,
```

```
            real, intent(out), optional dlnphidt,
            real, intent(out), optional dlnphidp )
```

This function calculates reduced deparure Gibbs energy, or the mixture fugacity coefficient. That is; equation 21 in chapter 2 of Michelsen book:

$$\frac{G(T,P,\mathbf{n})}{RT} = F(T,V,\mathbf{n}) + n\left(\frac{PV}{RT} - 1 - \ln(z)\right)$$

Author

Morten H.

Parameters

in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*tr*	Reduced temperature [-]
in	*pr*	Reduced pressure [-]
in	*vr*	Reduced volume [-]
in	*moles*	Total number of moles [mol]

Returns

The mixture fugacity coefficients [-]

Parameters

in	*simporref*	Simple (1) or reference (2) fluid

5.29.2.35 mainleekesler()

```
subroutine, public leekesler::mainleekesler (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc) comp,
            class(cb_eos), intent(in) cbeos,
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) nmoles,
            integer, intent(in) phase,
            real, intent(out) z,
            real, intent(out) sdep,
            real, intent(out) hdep,
            real, dimension(nc), intent(out) lnphi )
```

Main subroutine of the Lee Kesler eos. Takes temperature, pressure, composition and phase as input, and returns comrpessibility, entropy departure, enthalpy departure and fugacity coefficients. Calls upon routine thermProps to calculate these values for simple fluid (0) and reference fluid (1) seperatly, with reduced values for temperature and pressure calculated by the mixing rules, and combines the results according to the Lee Kesler relation:

$$M = M^{(0)} + \frac{\omega_M}{\omega^{(r)}}(M^{(r)} - M^{(0)})$$

where M is an arbitrary thermodynamic property.

Author

JA, 2013-07

Call routine with parameter 2, calculates for simple fluid.
Call routine with parameter 1, calculates for reference fluid.

5.29.2.36 mixrules()

```
subroutine leekesler::mixrules (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc) comp,
            class(cb_eos), intent(in) cbeos,
            real, intent(out) tcm,
            real, intent(out) vcm,
            real, intent(out) pcm,
            real, intent(out) zcm,
            real, intent(out) wm,
            real, dimension(nc), intent(in) nmoles,
            real, intent(in) moles )
```

Calculate the critical mixing temperature, volume, pressure, compressibility and acentricity factor, for given composition. Critical temperature and critical volume is used in the calculations, these values are known.
Mixing rules:

$$T_{cM} = \frac{1}{v_{cM}^{\eta} n^2} \sum_j \sum_k n_j n_k \cdot v_{c,jk}^{\eta} \cdot T_{c,jk}$$

$$v_{cM} = \frac{1}{n^2} \sum_j \sum_k n_j n_k \cdot v_{c,jk}$$

$$\omega_M = \frac{1}{n} \sum_j n_j \omega_j$$

$$z_{cM} = (0.2905 - 0.085\omega_M)$$

$$P_{cM} = R\frac{z_{cM} T_{cM}}{v_{cM}}$$

$$T_{c,jk} = (T_{c,j} \cdot T_{c,k})^{1/2} \cdot \kappa_{jk}$$

$$v_{c,jk} = \frac{1}{8}(v_{c,j}^{1/3} + v_{c,k}^{1/3})^3$$

Author

JA, 2013-06

5.29.2.37 pcmdiff2ninj()

```
real function leekesler::pcmdiff2ninj (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc) comp,
            class(cb_eos), intent(in) cbeos,
            real, intent(in) tcm,
            real, intent(in) vcm,
            real, intent(in) pcm,
            real, intent(in) zcm,
            real, intent(in) wm,
```

```
real, dimension(nc), intent(in) nmoles,
real, intent(in) moles,
integer, intent(in) i,
integer, intent(in) j )
```

Calculate the first order derivative of the pseudo critical mixing pressure, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.

$$\left(\frac{\partial^2 P_cM}{\partial n_i \partial n_j}\right) = \frac{1}{P_{cM}}\left(\frac{\partial P_{cM}}{\partial n_i}\right)\left(\frac{\partial P_{cM}}{\partial n_j}\right) + P_{cM}\Bigg[-\frac{1}{z_{cM}^2}\left(\frac{\partial z_{cM}}{\partial n_i}\right)\left(\frac{\partial z_{cM}}{\partial n_j}\right) + \frac{1}{z_{cM}}\left(\frac{\partial^2 z_cM}{\partial n_i \partial n_j}\right) - \frac{1}{T_{cM}^2}\left(\frac{\partial T_{cM}}{\partial n_i}\right)\left(\frac{\partial T_{cM}}{\partial n_j}\right) + \frac{1}{T_{cM}}\left(\frac{\partial^2 T_cM}{\partial n_i \partial n_j}\right) + \frac{1}{v_{cM}^2}\left(\frac{\partial v_{cM}}{\partial n_i}\right)\left(\frac{\partial v_{cM}}{\partial n_j}\right) - \frac{1}{v_{cM}}\left(\frac{\partial^2 v_cM}{\partial n_i \partial n_j}\right)\Bigg]$$

Author

JA, 2013-06

5.29.2.38 pcmdiffni()

```
real function leekesler::pcmdiffni (
              integer, intent(in) nc,
              type (gendata_pointer), dimension(nc) comp,
              class(cb_eos), intent(in) cbeos,
              real, intent(in) tcm,
              real, intent(in) vcm,
              real, intent(in) pcm,
              real, intent(in) zcm,
              real, intent(in) wm,
              real, dimension(nc), intent(in) nmoles,
              real, intent(in) moles,
              integer, intent(in) i )
```

Calculate the first order derivative of the pseudo critical mixing pressure, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.

$$\left(\frac{\partial P_{cM}}{\partial n_i}\right)_{T,V} = P_{cM}\left(\frac{1}{z_{cM}}\left(\frac{\partial z_{cM}}{\partial n_i}\right)_{T,V} + \frac{1}{T_{cM}}\left(\frac{\partial T_{cM}}{\partial n_i}\right)_{T,V} - \frac{1}{v_{cM}}\left(\frac{\partial v_{cM}}{\partial n_i}\right)_{T,V}\right)$$

Author

JA, 2013-06

5.29.2.39 pdiffni()

```
real function leekesler::pdiffni (
              integer, intent(in) nc,
              type (gendata_pointer), dimension(nc) comp,
              class(cb_eos), intent(in) cbeos,
              real, intent(in) moles,
              real, intent(in) tr,
              real, intent(in) vr,
              real, intent(in) tcm,
              real, intent(in) vcm,
              real, intent(in) pcm,
```

```
            real, intent(in) zcm,
            real, intent(in) wm,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            real, dimension(nc), intent(in) nmoles,
            integer, intent(in) i,
            integer, intent(in) simporref )
```

Calculate the derivative of the pressure, with respect to composition, for fixed temperature and volume.

$$\left(\frac{\partial P}{\partial n_i}\right)_{T,V} = -RT\left(\frac{\partial^2 F}{\partial V \partial n_i}\right)_T + \frac{RT}{V}$$

Author

JA, 2013-06

5.29.2.40 pdifft()

```
real function leekesler::pdifft (
            real, intent(in) t,
            real, intent(in) p,
            real, intent(in) moles,
            real, intent(in) tr,
            real, intent(in) vr,
            real, intent(in) tcm,
            real, intent(in) pcm,
            integer, intent(in) simporref )
```

Pressure P = P(T,V,n) Derivatives of P with respect to temperature, volume and composition are used when finding several of the derivatives of the thermodynamic properties of interest. The functions for these derivatives follow. Calculate the derivative of the pressure, with respect to temperature, for fixed volume and composition.

$$\left(\frac{\partial P}{\partial T}\right)_{V,\mathbf{n}} = \frac{P}{T} - RT\left(\frac{\partial^2 F}{\partial T \partial V}\right)_{n_i}$$

Author

JA, 2013-06

5.29.2.41 pdiffv()

```
real function leekesler::pdiffv (
            real, intent(in) moles,
            real, intent(in) tr,
            real, intent(in) vr,
            real, intent(in) tcm,
            real, intent(in) pcm,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            integer, intent(in) simporref )
```

Calculate the derivative of the pressure, with respect to volume, for fixed temperature and composition.

$$\left(\frac{\partial P}{\partial V}\right)_{T,\mathbf{n}} = -RT\left(\frac{\partial^2 F}{\partial V^2}\right)_{T,\mathbf{n}} - \frac{nRT}{V^2}$$

Author

JA, 2013-06

5.29.2.42 pred()

```
real function leekesler::pred (
             real, intent(in) tr,
             real, intent(in) vr,
             integer, intent(in) simporref )
```

Calculate the reduced pressure given reduzed temperature and reduced volume:

$$Pr = \frac{T_r}{v_r}\left[1 - \frac{v_r}{n}\left(\frac{\partial F}{\partial v_r}\right)_{T_r,n}\right]$$

Author

MH, 2013-09

5.29.2.43 prsolver()

```
real function leekesler::prsolver (
             real, intent(in) tr,
             real, intent(in) vr,
             real, intent(in) b,
             real, intent(in) c,
             real, intent(in) d,
             real, intent(in) e,
             integer, intent(in) simporref )
```

Calculate the reduced pressure, Pr, by directly solving the Lee Keslers equation of state, given reduced temperature, reduced volume and composition.

Author

JA, 2013-07

5.29.2.44 sdiffni()

```
real function leekesler::sdiffni (
             integer, intent(in) nc,
             type (gendata_pointer), dimension(nc) comp,
             class(cb_eos), intent(in) cbeos,
             real, intent(in) t,
             real, intent(in) p,
             real, intent(in) moles,
             real, intent(in) tr,
             real, intent(in) vr,
             real, intent(in) tcm,
             real, intent(in) vcm,
             real, intent(in) pcm,
             real, intent(in) zcm,
             real, intent(in) wm,
             real, intent(in) b,
             real, intent(in) c,
             real, intent(in) d,
             real, intent(in) e,
             real, dimension(nc), intent(in) nmoles,
             integer, intent(in) i,
```

```
            real, intent(in) z,
            integer, intent(in) simporref )
```

Calculate the derivative of entropy, with respect to pressure, for fixed temperature and composition.

$$\left(\frac{\partial S^R(T,P,\mathbf{n})}{\partial n_i}\right)_{T,P} = \bar{V}_i \left(\frac{\partial P}{\partial T}\right)_{V,\mathbf{n}} - R\left[\left(\frac{\partial F}{\partial n_i}\right)_{T,V} + T\left(\frac{\partial^2 F}{\partial T \partial n_i}\right)_V + 1 - \ln z\right]$$

Author

JA, 2013-06

5.29.2.45 sdiffp()

```
real function leekesler::sdiffp (
            real, intent(in) t,
            real, intent(in) p,
            real, intent(in) moles,
            real, intent(in) tr,
            real, intent(in) vr,
            real, intent(in) tcm,
            real, intent(in) pcm,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            integer, intent(in) simporref )
```

Calculate the derivative of entropy, with respect to pressure, for fixed temperature and composition.

$$\left(\frac{\partial S^R(T,P,\mathbf{n})}{\partial P}\right)_{T,\mathbf{n}} = \frac{nR}{P} - \bar{V}_T$$

Author

JA, 2013-06

5.29.2.46 sdifft()

```
real function leekesler::sdifft (
            real, intent(in) t,
            real, intent(in) p,
            real, intent(in) moles,
            real, intent(in) tr,
            real, intent(in) vr,
            real, intent(in) tcm,
            real, intent(in) pcm,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            integer, intent(in) simporref )
```

The derivatives of the entropy, with respect to temperature, pressure and composition follow.
Calculate the derivative of entropy, with respect to temperature, for fixed pressure and composition.

$$\left(\frac{\partial S^R(T,P,\mathbf{n})}{\partial T}\right)_{P,\mathbf{n}} = \bar{V}_T \left(\frac{\partial P}{\partial T}\right)_{V,\mathbf{n}} - R\left[2\left(\frac{\partial F}{\partial T}\right)_{V,\mathbf{n}} + T\left(\frac{\partial^2 F}{\partial T^2}\right)_{V,\mathbf{n}} + \frac{n}{T}\right]$$

Author

JA, 2013-06

5.29.2.47 setmaxminz()

```
subroutine leekesler::setmaxminz (
             real, intent(in) tr,
             real, intent(in) pr,
             integer, intent(in) phase,
             real, intent(out) z,
             real, intent(out) zmax,
             real, intent(out) zmin )
```

Set limits for compressibillity factor, and initial value based on phase flag.

Author

MH, 2013-09

5.29.2.48 tcmdiff2ninj()

```
real function leekesler::tcmdiff2ninj (
             integer, intent(in) nc,
             type (gendata_pointer), dimension(nc) comp,
             class(cb_eos), intent(in) cbeos,
             real, intent(in) tcm,
             real, intent(in) vcm,
             real, dimension(nc), intent(in) nmoles,
             real, intent(in) moles,
             integer, intent(in) i,
             integer, intent(in) j )
```

Calculate the second order derivative of critical mixing temperature, with respect to composition, for fixed temperature and volume. Help function to be used in finding the second order derivative of F, with respect to composition.

$$\begin{aligned}\left(\frac{\partial^2 T_cM}{\partial n_i \partial n_j}\right) =& \frac{2}{n^2}T_{cM} - \frac{2}{n}\left(\frac{\partial T_{cM}}{\partial n_j}\right) - \frac{\eta}{v_{cM}}\left(\frac{\partial v_{cM}}{\partial n_i}\right)\left(\frac{\partial T_{cM}}{\partial n_j}\right) \\ &+ \frac{\eta T_{cM}}{v_{cM}^2}\left(\frac{\partial v_{cM}}{\partial n_i}\right)\left(\frac{\partial v_{cM}}{\partial n_j}\right) - \frac{\eta T_{cM}}{v_{cM}}\left(\frac{\partial^2 v_cM}{\partial n_i \partial n_j}\right) \\ &- \frac{4}{n^3 v_{cM}^{\eta}}\sum_l n_l v_{c,il}^{\eta} T_{c,il} - \frac{2\eta}{v_{cM}^{\eta+1}n^2}\left(\frac{\partial v_{cM}}{\partial n_j}\right)\sum_l n_l v_{c,il}^{\eta} T_{c,il} \\ &+ \frac{2}{v_{cM}^{\eta} n^2} v_{c,ij}^{\eta} T_{c,ij}\end{aligned}$$

Author

JA, 2013-06

5.29.2.49 tcmdiffni()

```
real function leekesler::tcmdiffni (
             integer, intent(in) nc,
             type (gendata_pointer), dimension(nc) comp,
             class(cb_eos), intent(in) cbeos,
             real, intent(in) tcm,
             real, intent(in) vcm,
             real, dimension(nc), intent(in) nmoles,
             real, intent(in) moles,
             integer, intent(in) i )
```

Calculate the first order derivative of critical mixing temperature, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.

$$\left(\frac{\partial T_{cM}}{\partial n_i}\right)_{T,V} = -\frac{2}{n}T_{cM} - \frac{\eta T_{cM}}{v_{cM}}\left(\frac{\partial v_{cM}}{\partial n_i}\right)_{T,V} + \frac{2}{v_{cM}^{\eta} n^2}\sum_l n_l v_{c,il}^{\eta} T_{c,il}$$

Author

JA, 2013-06

5.29.2.50 testdiffleekesler()

```
subroutine, public leekesler::testdiffleekesler (
             integer, intent(in) nc,
             type (gendata_pointer), dimension(nc) comp,
             class(cb_eos), intent(in) cbeos,
             real, intent(in) tin,
             real, intent(in) pin,
             real, dimension(nc), intent(in) z,
             integer, intent(in) phase )
```

Test differentials of the Lee Kesler eos. Takes temperature, pressure, composition and phase as input, and test compressibility, entropy departure, enthalpy departure and fugacity coefficients differentials.

Author

Morten H, 2013-09

5.29.2.51 thermprops()

```
subroutine leekesler::thermprops (
             integer, intent(in) nc,
             type (gendata_pointer), dimension(nc) comp,
             class(cb_eos), intent(in) cbeos,
             real, intent(in) tr,
             real, intent(in) pr,
             real, intent(in) vr,
             real, dimension(nc), intent(in) nmoles,
             real, intent(in) tcm,
             real, intent(in) vcm,
             real, intent(in) pcm,
             real, intent(in) zcm,
             real, intent(in) wm,
             real, intent(in) moles,
             real, intent(out) z,
             real, intent(out) s,
             real, intent(out) h,
             real, dimension(nc), intent(out) lnphi,
             integer, intent(in) simporref )
```

Subroutine to calculate the thermodynamic properties of either simple fluid or reference fluid. Solutions are compined in main routine.
Thermodynamic properties calculated by:

$$z = \frac{P_r v_r}{T_r}$$
$$S^R(T,V,\mathbf{n}) = -R\left[F + T\left(\frac{\partial F}{\partial T}\right)_{V,\mathbf{n}}\right]$$
$$H^R(T,P,\mathbf{n}) = RT\left(\frac{\partial F}{\partial T}\right)_{V,\mathbf{n}} + PV - nRT$$
$$\ln\phi_i(T,P,\mathbf{n}) = \left(\frac{\partial F}{\partial n_i}\right)_{T,V} - \ln z$$

Author

JA, 2013-07

5.29.2.52 trcoeff()

```
subroutine leekesler::trcoeff (
             real, intent(in) tr,
             real, intent(out) b,
             real, intent(out) c,
             real, intent(out) d,
             real, intent(out) e,
             integer, intent(in) simporref )
```

Calculate the reduced temperature dependent coefficients B,C,D and E, in the expression for Helmholts reduced recidual function F, for given reduced temperature.

$$B(T_r) = b_1 - \frac{b_2}{T_r} - \frac{b_3}{T_r^2} - \frac{b_4}{T_r^3}$$
$$C(T_r) = c_1 - \frac{c_2}{T_r} + \frac{c_3}{T_r^3}$$
$$D(T_r) = d_1 + \frac{d_2}{T_r}$$
$$E(T_r) = \frac{c_4}{T_r^3}$$

Author

JA, 2013-06

5.29.2.53 trcoeffdiff1()

```
subroutine leekesler::trcoeffdiff1 (
             real, intent(in) tr,
             real, intent(out) b_tr,
             real, intent(out) c_tr,
             real, intent(out) d_tr,
             real, intent(out) e_tr,
             integer, intent(in) simporref )
```

Calculate the first order derivatives of the reduced temperature dependent coefficients B,C,D and E, for given reduced temperature.

$$B_{T_r} = \frac{\partial B}{\partial T_r} = \frac{b_2}{T_r^2} + \frac{2b_3}{T_r^3} + \frac{3b_4}{T_r^4}$$
$$C_{T_r} = \frac{\partial C}{\partial T_r} = \frac{c_2}{T_r^2} - \frac{3c_3}{T_r^4}$$
$$D_{T_r} = \frac{\partial D}{\partial T_r} = -\frac{d_2}{T_r^2}$$
$$E_{T_r} = \frac{\partial E}{\partial T_r} = -\frac{3c_4}{T_r^4}$$

Author

JA, 2013-06

5.29.2.54 trcoeffdiff2()

```
subroutine leekesler::trcoeffdiff2 (
            real, intent(in) tr,
            real, intent(out) b_trtr,
            real, intent(out) c_trtr,
            real, intent(out) d_trtr,
            real, intent(out) e_trtr,
            integer, intent(in) simporref )
```

Calculate the second order derivatives of the reduced temperature dependent coefficients B,C,D and E, for given reduced temperature.

$$B_{TT_r} = \frac{\partial^2 B}{\partial^2 T_r} = -\frac{2b_2}{T_r^3} - \frac{6b_3}{T_r^4} - \frac{12b_4}{T_r^5}$$
$$C_{TT_r} = \frac{\partial^2 C}{\partial^2 T_r} = -\frac{2c_2}{T_r^3} + \frac{12c_3}{T_r^5}$$
$$D_{TT_r} = \frac{\partial^2 D}{\partial^2 T_r} = \frac{2d_2}{T_r^3}$$
$$E_{TT_r} = \frac{\partial^2 E}{\partial^2 T_r} = \frac{12c_4}{T_r^5}$$

Author

JA, 2013-06

5.29.2.55 trdiff2ninj()

```
real function leekesler::trdiff2ninj (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc) comp,
            class(cb_eos), intent(in) cbeos,
            real, intent(in) tr,
            real, intent(in) tcm,
            real, intent(in) vcm,
            real, dimension(nc), intent(in) nmoles,
            real, intent(in) moles,
            integer, intent(in) i,
            integer, intent(in) j )
```

Calculate the second order derivative of reduced temperature, with respect to composition, for fixed temperature and volume. Help function to be used in finding the second order derivative of F, with respect to composition.

$$X_{ij} = \left(\frac{\partial^2 T_r}{\partial n_i \partial n_j}\right)_{T,V} = \frac{2T_r}{T_{cM}^2}\left(\frac{\partial T_{cM}}{\partial n_i}\right)\left(\frac{\partial T_{cM}}{\partial n_j}\right) - \frac{T_r}{T_{cM}}\left(\frac{\partial^2 T_c M}{\partial n_i \partial n_j}\right)$$

Author

JA, 2013-06

5.29.2.56 trdiffni()

```
real function leekesler::trdiffni (
              integer, intent(in) nc,
              type (gendata_pointer), dimension(nc) comp,
              class(cb_eos), intent(in) cbeos,
              real, intent(in) tr,
              real, intent(in) tcm,
              real, intent(in) vcm,
              real, dimension(nc), intent(in) nmoles,
              real, intent(in) moles,
              integer, intent(in) i )
```

Calculate the first order derivative of reduced temperature, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.

$$X_i = \left(\frac{\partial T_r}{\partial n_i}\right)_{T,V} = -\frac{T_r}{T_{cM}}\left(\frac{\partial T_{cM}}{\partial n_i}\right)_{T,V}$$

Author

JA, 2013-06

5.29.2.57 vcmdiff2ninj()

```
real function leekesler::vcmdiff2ninj (
              integer, intent(in) nc,
              type (gendata_pointer), dimension(nc) comp,
              real, dimension(nc), intent(in) nmoles,
              real, intent(in) moles,
              real, intent(in) vcm,
              integer, intent(in) i,
              integer, intent(in) j )
```

Calculate the second order derivative of critical mixing volume, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.

$$\begin{aligned}\left(\frac{\partial^2 v_c M}{\partial n_i \partial n_j}\right) &= \frac{2}{n^2} v_{cM} - \frac{2}{n}\left(\frac{\partial v_{cM}}{\partial n_j}\right) - \frac{4}{n^3}\sum_l n_l v_{c,il} + \frac{2}{n^2} v_{c,ij} \\ &= \frac{2}{n^2}(v_{c,ij} - v_{cM}) - \frac{2}{n}\left[\left(\frac{\partial v_{cM}}{\partial n_i}\right) + \left(\frac{\partial v_{cM}}{\partial n_j}\right)\right]\end{aligned}$$

Author

JA, 2013-06

5.29.2.58 vcmdiffni()

```
real function leekesler::vcmdiffni (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc) comp,
            real, dimension(nc), intent(in) nmoles,
            real, intent(in) vcm,
            real, intent(in) moles,
            integer, intent(in) i )
```

Calculate the first order derivative of critical mixing volume, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.

$$\left(\frac{\partial v_{cM}}{\partial n_i}\right)_{T,V} = -\frac{2}{n}v_{cM} + \frac{2}{n^2}\sum_l n_l v_{c,il}$$

Author

JA, 2013-06

5.29.2.59 vdiffni()

```
real function leekesler::vdiffni (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc) comp,
            class(cb_eos), intent(in) cbeos,
            real, intent(in) moles,
            real, intent(in) tr,
            real, intent(in) vr,
            real, intent(in) tcm,
            real, intent(in) vcm,
            real, intent(in) pcm,
            real, intent(in) zcm,
            real, intent(in) wm,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            real, dimension(nc), intent(in) nmoles,
            integer, intent(in) i,
            integer, intent(in) simporref )
```

Calculate the derivative of the volume, with respect to composition, For fixed temperature and pressure. Help function that simplifies notation.

$$\bar{V}_i \equiv \left(\frac{\partial V}{\partial n_i}\right)_{T,P} = -\frac{\left(\frac{\partial P}{\partial n_i}\right)_{T,V}}{\left(\frac{\partial P}{\partial V}\right)_{T,\mathbf{n}}}$$

Author

JA, 2013-06

5.29.2.60 vdifft()

```
real function leekesler::vdifft (
            real, intent(in) t,
            real, intent(in) p,
            real, intent(in) moles,
            real, intent(in) tr,
```

```
            real, intent(in) vr,
            real, intent(in) tcm,
            real, intent(in) pcm,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            integer, intent(in) simporref )
```

Calculate the derivative of the volume, with respect to temperature, For fixed pressire and composition. Help function that simplifies notation.

$$\bar{V}_T \equiv \left(\frac{\partial V}{\partial T}\right)_{P,\mathbf{n}} = -\frac{\left(\frac{\partial P}{\partial T}\right)_{V,\mathbf{n}}}{\left(\frac{\partial P}{\partial V}\right)_{T,\mathbf{n}}}$$

Author

JA, 2013-06

5.29.2.61 vrdiff2ninj()

```
real function leekesler::vrdiff2ninj (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc) comp,
            class(cb_eos), intent(in) cbeos,
            real, intent(in) vr,
            real, intent(in) tcm,
            real, intent(in) vcm,
            real, intent(in) pcm,
            real, intent(in) zcm,
            real, intent(in) wm,
            real, dimension(nc), intent(in) nmoles,
            real, intent(in) moles,
            integer, intent(in) i,
            integer, intent(in) j )
```

Calculate the second order derivative of reduced volume, with respect to composition, for fixed temperature and volume. Help function to be used in finding the second order derivative of F, with respect to composition.

$$\begin{aligned} Y_{ij} = \left(\frac{\partial^2 v_r}{\partial n_i \partial n_j}\right)_{T,V} = \frac{1}{v_r}\left(\frac{\partial v_r}{\partial n_i}\right)_{T,V}\left(\frac{\partial v_r}{\partial n_j}\right)_{T,V} - \frac{v_r}{P_{cM}^2}\left(\frac{\partial P_{cM}}{\partial n_i}\right)\left(\frac{\partial P_{cM}}{\partial n_j}\right) \\ + \frac{v_r}{P_{cM}}\left(\frac{\partial^2 P_c M}{\partial n_i \partial n_j}\right) + \frac{v_r}{T_{cM}^2}\left(\frac{\partial T_{cM}}{\partial n_i}\right)\left(\frac{\partial T_{cM}}{\partial n_j}\right) \\ - \frac{v_r}{T_{cM}}\left(\frac{\partial^2 T_c M}{\partial n_i \partial n_j}\right) + \frac{v_r}{n^2} \end{aligned}$$

Author

JA, 2013-06

5.29.2.62 vrdiffni()

```
real function leekesler::vrdiffni (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc) comp,
            class(cb_eos), intent(in) cbeos,
```

```
            real, intent(in) vr,
            real, intent(in) tcm,
            real, intent(in) vcm,
            real, intent(in) pcm,
            real, intent(in) zcm,
            real, intent(in) wm,
            real, dimension(nc), intent(in) nmoles,
            real, intent(in) moles,
            integer, intent(in) i )
```

Calculate the first order derivative of reduced volume, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.

$$Y_i = \left(\frac{\partial v_r}{\partial n_i}\right)_{T,V} = \frac{v_r}{P_{cM}}\left(\frac{\partial P_{cM}}{\partial n_i}\right)_{T,V} - \frac{v_r}{T_{cM}}\left(\frac{\partial T_{cM}}{\partial n_i}\right)_{T,V} - \frac{v_r}{n}$$

Author

JA, 2013-06

5.29.2.63 vrinitial()

```
subroutine leekesler::vrinitial (
            real, intent(in) pr,
            real, intent(in) tr,
            integer, intent(in) usedphase,
            integer, intent(in) simporref,
            real, intent(out) vrinit,
            real, intent(out) vrmin,
            real, intent(out) vrmax )
```

The following subroutine is a routine that helps solving the non-linear equation of state (i.e. finding the reduced volume) by calculationg an appropriate initial value to be used in the numerical method. A bound for the allowed value of the reduced specific volume is also found.

Author

Ander, 2007, adapted by JA, 2013-07

5.29.2.64 vrnewtraps()

```
subroutine leekesler::vrnewtraps (
            real, intent(in) tr,
            real, intent(in) pr,
            integer, intent(in) usedphase,
            integer, intent(in) simporref,
            real, intent(out) vr,
            logical, intent(out) correctphase )
```

Calculate the reduced volume, vr, by use of the Newton-Rapson numerical method to solve Lee Keslers equation of state, given reduced temperature, reduced pressure and composition.The half step method is used to secure that vr is inside its limit. Calls upon functions fv and fvDiff, acting as the function f and the derivative of f in the numerical method.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Author

JA, 2013-07

5.29.2.65 wmdiff2ninj()

```
real function leekesler::wmdiff2ninj (
             integer, intent(in) nc,
             type (gendata_pointer), dimension(nc) comp,
             real, intent(in) moles,
             real, intent(in) wm,
             integer, intent(in) i,
             integer, intent(in) j )
```

Calculate the second order derivative of the pseudo critical acentricity factor, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.

$$\left(\frac{\partial^2 \omega_M}{\partial n_i \partial n_j}\right) = -\frac{1}{n}\left[\left(\frac{\partial \omega_M}{\partial n_i}\right) + \left(\frac{\partial w_M}{\partial n_j}\right)\right]$$

Author

JA, 2013-06

5.29.2.66 wmdiffni()

```
real function leekesler::wmdiffni (
             integer, intent(in) nc,
             type (gendata_pointer), dimension(nc) comp,
             real, intent(in) moles,
             real, intent(in) wm,
             integer, intent(in) i )
```

Calculate the first order derivative of the pseudo critical acentricity factor, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.

$$\left(\frac{\partial \omega_M}{\partial n_i}\right)_{T,V} = \frac{1}{n}(\omega_i - \omega_M)$$

Author

JA, 2013-06

5.29.2.67 zcmdiff2ninj()

```
real function leekesler::zcmdiff2ninj (
             integer, intent(in) nc,
             type (gendata_pointer), dimension(nc) comp,
             real, intent(in) moles,
             real, intent(in) wm,
             integer, intent(in) i,
             integer, intent(in) j )
```

Calculate the second order derivative of the pseudo critical mixing compressibility, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.

$$\left(\frac{\partial^2 z_c M}{\partial n_i \partial n_j}\right) = -0.085\left(\frac{\partial^2 \omega_M}{\partial n_i \partial n_j}\right)$$

Author

JA, 2013-06

5.29.2.68 zcmdiffni()

```
real function leekesler::zcmdiffni (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc) comp,
            real, intent(in) moles,
            real, intent(in) wm,
            integer, intent(in) i )
```

Calculate the first order derivative of the pseudo critical mixing compressibility, with respect to composition, for fixed temperature and volume. Help function to be used in finding the derivative of F, with respect to composition.

$$\left(\frac{\partial z_{cM}}{\partial n_i}\right)_{T,V} = -0.085\left(\frac{\partial \omega_M}{\partial n_i}\right)_{T,V}$$

Author

JA, 2013-06

5.29.2.69 zdiffni()

```
real function leekesler::zdiffni (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc) comp,
            class(cb_eos), intent(in) cbeos,
            real, intent(in) z,
            real, intent(in) moles,
            real, intent(in) tr,
            real, intent(in) vr,
            real, intent(in) tcm,
            real, intent(in) vcm,
            real, intent(in) pcm,
            real, intent(in) zcm,
            real, intent(in) wm,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            real, dimension(nc), intent(in) nmoles,
            integer, intent(in) i,
            integer, intent(in) simporref )
```

Calculate the derivative of the compressibility, with respect to composition, for fixed temperature and pressure.

$$\left(\frac{\partial z}{\partial n_i}\right)_{T,P} = -z\left[\frac{1}{n} - \frac{\bar{V}_i}{V}\right]$$

Author

JA, 2013-06

5.29.2.70 zdiffp()

```
real function leekesler::zdiffp (
            real, intent(in) z,
            real, intent(in) p,
            real, intent(in) moles,
            real, intent(in) tr,
            real, intent(in) vr,
            real, intent(in) tcm,
```

```
            real, intent(in) pcm,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            integer, intent(in) simporref )
```

Calculate the derivative of the compressibility, with respect to pressure, for fixed temperature and composition.

$$\left(\frac{\partial z}{\partial P}\right)_{T,\mathbf{n}} = z\left[\frac{1}{P} + \frac{1}{V\left(\frac{\partial P}{\partial V}\right)_{T,\mathbf{n}}}\right]$$

Author

JA, 2013-06

5.29.2.71 zdifft()

```
real function leekesler::zdifft (
            real, intent(in) z,
            real, intent(in) t,
            real, intent(in) p,
            real, intent(in) moles,
            real, intent(in) tr,
            real, intent(in) vr,
            real, intent(in) tcm,
            real, intent(in) pcm,
            real, intent(in) b,
            real, intent(in) c,
            real, intent(in) d,
            real, intent(in) e,
            integer, intent(in) simporref )
```

The derivatives of the compressibility, with respect to temperature, pressure and composition follow.
Calculate the derivative of the compressibility, with respect to temperature, for fixed pressure and composition.

$$\left(\frac{\partial z}{\partial T}\right)_{P,\mathbf{n}} = -z\left[\frac{1}{T} - \frac{\bar{V}_T}{V}\right]$$

Author

JA, 2013-06

5.29.2.72 zinitial()

```
subroutine leekesler::zinitial (
            real, intent(in) tr,
            real, intent(in) pr,
            integer, intent(in) phase,
            integer, intent(in) simporref,
            real, intent(out) z,
            real, intent(out) zmin,
            real, intent(out) zmax,
            logical, intent(out) solved,
            logical, intent(out) hasphase )
```

Find phase minima/maxima of the Lee Keslers equation of state. If f(z) = 0 have a solution an interval for the solution is returned. If the search don't cross f(z) = 0, the minima/maxima is returned. Calls upon functions fz, fzDiff and fzDiff2 acting as the function f, derivative of f and second derivative in the numerical method.

$$z_{n+1} = z_n - \frac{f'(z_n)}{f''(x_n)}$$

Author

MH, 2013-09

5.29.2.73 znewtraps()

```
subroutine leekesler::znewtraps (
             real, intent(in) tr,
             real, intent(in) pr,
             integer, intent(in) phase,
             integer, intent(in) simporref,
             real, intent(out) vr,
             real, intent(in) zinit,
             real, intent(inout) zmax,
             real, intent(inout) zmin )
```

Calculate the reduced volume, vr, by use of the Newton-Rapson numerical method to solve Lee Keslers equation of state, given reduced temperature, reduced pressure and composition.The half step method is used to secure that vr is inside its limit. Calls upon functions fz, fzDiff and fzDiff2 acting as the function f, derivative of f and second derivative in the numerical method.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \left(1 + \frac{f(x_n)f''(x_n)}{2f'(x_n)^2}\right)$$

A Taylor expansion of f is used, and the root giving the smallest change in x is used. To simplify the step, the following series approximation is used,

$$1 - \sqrt{1-\alpha} = \frac{\alpha}{2} + \frac{\alpha^2}{8} + O(\alpha^3).$$

Author

MH, 2013-09

5.29.2.74 zprtshape()

```
subroutine leekesler::zprtshape (
             real, intent(in) t,
             real, intent(in) tr,
             real, intent(in) b,
             real, intent(in) c,
             real, intent(in) d,
             real, intent(in) e,
             integer, intent(in) simporref )
```

Optional subroutines and functions. These arenot called upon automatically, hence the thermProps routine should be modified to call upon them, if this output is desired.
The two following subroutines are help routines to generate neat output of large amounts of data. Output is used to get an intuitive understanding of the shape of the compressibility z and the function fv (of which the roots are found by the Newton-Rapson method), respectfully.

Author

JA, 2013-07

5.30 linear_numerics Module Reference

This module contains methods for solving systems of linear equations.

Functions/Subroutines

- subroutine, public cg (a, x, b)

  A simple CG solver, can be used instead of LAPACK's dgesv for solving linear systems.
- subroutine, public **solvelu** (neq, x, a, symmetric, ierr)
- subroutine, public inverse (matrix, inv, n, errorflag)

  Subroutine to find the inverse of a square matrix Modified after freely available routine written by Ashwith J. Rego.
- subroutine, public null_space (a, n, x_null, ierr)

  Calculate null space of A using SVD.
- subroutine, public outer_product (a, na, b, nb, ab)

  Calculate outer product.
- subroutine, public solve_lu_hd (a, n, b, ierr)

  Solve a x = b.

5.30.1 Detailed Description

This module contains methods for solving systems of linear equations.

Author

MH, August 2022

5.30.2 Function/Subroutine Documentation

5.30.2.1 cg()

```
subroutine, public linear_numerics::cg (
             real, dimension(:,:), intent(in) a,
             real, dimension(:), intent(inout) x,
             real, dimension(:), intent(in) b )
```

A simple CG solver, can be used instead of LAPACK's dgesv for solving linear systems.

Author

KEGT

5.30.2.2 inverse()

```
subroutine, public linear_numerics::inverse (
             real, dimension(n,n), intent(in) matrix,
             real, dimension(n,n), intent(out) inv,
             integer, intent(in) n,
             integer, intent(out) errorflag )
```

Subroutine to find the inverse of a square matrix Modified after freely available routine written by Ashwith J. Rego.

Author

MH, August 2012

5.30.2.3 null_space()

```
subroutine, public linear_numerics::null_space (
             real, dimension(n,n), intent(inout) a,
             integer, intent(in) n,
             real, dimension(n), intent(inout) x_null,
             integer, intent(out) ierr )
```

Calculate null space of A using SVD.

Author

MH, August 2022

Parameters

in,out	*a*	Symmetric matrix
in	*n*	Dimension of A
in,out	*x_null*	Null space of A
out	*ierr*	Error flag

5.30.2.4 outer_product()

```
subroutine, public linear_numerics::outer_product (
             real, dimension(na), intent(in) a,
             integer, intent(in) na,
             real, dimension(nb), intent(in) b,
             integer, intent(in) nb,
             real, dimension(na,nb), intent(out) ab )
```

Calculate outer product.

Author

MH, August 2022

5.30.2.5 solve_lu_hd()

```
subroutine, public linear_numerics::solve_lu_hd (
             type(hyperdual), dimension(n,n), intent(inout) a,
             integer, intent(in) n,
             type(hyperdual), dimension(n), intent(inout) b,
             integer, intent(out) ierr )
```

Solve a x = b.

Author

MH, January 2023

5.31 mbwr Module Reference

MBWR module.

Data Types

- type eosmbwr

  MBWR model type for mbwr19 and mbwr32.
- type nijlarray

Functions/Subroutines

- subroutine **allocnijl** (nijl, len)
- subroutine **deallocnijl** (nijl)
- subroutine **dealloceosmbwr** (refeosmbwr)
- subroutine **nijlassign** (nijl, idx, nn, ii, jj, ll)
- subroutine initializembwrmodel (compid, model, nineteenor32)
- subroutine **readdbparameters** (compid, model, nineteenor32)
- subroutine **fillijl32** (nijl)
- subroutine **fillijl19** (nijl)
- subroutine **computezcoeff** (model)
- subroutine **computehelmcoeff** (model)

- subroutine **mbwr_coef** (ntderivatives, t, rhocoef, model)
- subroutine **makeparam** (parameters, t, model, ntderivatives)
- subroutine **mbwr_pressure** (rho, param, p, dpdrho, d2pdrho2)
- logical function **densityrootdoesntexist** (redt, phase, rho_old, rho_new, prho_old, prho, dpdrho, iter)

  Helper function for the search for density root in newton_density.
- subroutine newton_density (fun, param, redt, p, rho_inout, phase_in, rho_releps, p_releps, meta_extrem, prho_init, dpdrho_init)

  A Newton solver targeted at solving for density from a function P = P(rho). fun is a function handle computing pressure derivatives, having the form fun(x, param, p, dpdrho, d2pdrho2) where the last three arguments are optional.
- real function mbwr_density (t, p, phase_in, param, model, phase_found_out, meta_extrem)

  Interface to the density solver. Outputs density [mol/L].
- real function **mbwr_density_tplib** (t, p, phase, rhocoef, model)

  A reimplementation of the tplib-solver, where SRK is used to find the initial guess.
- real function **mb_frt** (d, rhocoef, model)
- real function **liquiddensitysrkguess** (t, p, model)
- logical function **extremasearchisdiverging** (phase, drho, prho_old, prho, dpdrho, p_in)

  Helper function for find_extremum.
- subroutine find_extremum (fun, p_in, phase, param, found_extrem, rho_extrem)

  Only works as a subroutine for newton_density, as the algorithm assumes that no root exists in the given phase.
- subroutine **mbwr_criticalparameters** (tc, pc, rc, model)
- real function **barenewton** (fun, param, p_in, x, x_releps, f_releps)

Variables

- integer, parameter **bplen19** = 6
- integer, parameter **belen19** = 2

  the number of coefficients of the rho-powers in the polynomial part and exponential part of MBWR-19
- integer, parameter **bplen32** = 9
- integer, parameter **belen32** = 6

  the number of coefficients of the rho-powers in the polynomial part and exponential part of MBWR-32
- integer, parameter **ipol19** = 13
- integer, parameter **iexp19** = 6

  the number of fitted parameters in the polynomial part and exponential part of MBWR-19 (not counting gamma)
- integer, parameter **ipol32** = 19
- integer, parameter **iexp32** = 13

  the number of fitted parameters in the polynomial part and exponential part of MBWR-32 (not counting gamma)
- real, parameter **pi** = 4.e0∗ATAN(1.0)
- logical **verbose** = .false.

5.31.1 Detailed Description

MBWR module.

5.31.2 Function/Subroutine Documentation

5.31.2.1 find_extremum()

```
subroutine mbwr::find_extremum (
            external subroutine(real, intent(in) rho, real, dimension(:), intent(in) param,
real, intent(out) p, real, intent(out), optional dpdrho, real, intent(out), optional d2pdrho2)
fun,
            real, intent(in) p_in,
            integer, intent(in) phase,
            real, dimension(:), intent(in) param,
            logical, intent(out) found_extrem,
            real, intent(inout) rho_extrem )
```

Only works as a subroutine for newton_density, as the algorithm assumes that no root exists in the given phase.

Parameters

in	*p_in*	Pressure [Pa]
in	*phase*	Desired phase
in	*param*	The parameters fun depends on
in,out	*rho_extrem*	Comes in with the initial value, comes out with metastable density

5.31.2.2 initializembwrmodel()

```
subroutine mbwr::initializembwrmodel (
            character(len=*), intent(in) compid,
            type(eosmbwr), intent(inout) model,
            integer, intent(in) nineteenor32 )
```

Parameters

in	*compid*	e.g. C3
in	*nineteenor32*	19 or 32

5.31.2.3 mbwr_density()

```
real function mbwr::mbwr_density (
            real, intent(in) t,
            real, intent(in) p,
            integer, intent(in) phase_in,
            real, dimension(:), intent(in) param,
            type(eosmbwr), intent(in) model,
            integer, intent(out) phase_found_out,
            logical, intent(in), optional meta_extrem )
```

Interface to the density solver. Outputs density [mol/L].

Parameters

in	*t*	Temperature [K]
in	*p*	Pressure [Pa]

5.31.2.4 newton_density()

```
subroutine mbwr::newton_density (
            external subroutine(real, intent(in) rho, real, dimension(:), intent(in) param,
real, intent(out) p, real, intent(out), optional dpdrho, real, intent(out), optional d2pdrho2)
fun,
            real, dimension(:), intent(in) param,
            real, intent(in) redt,
            real, intent(in) p,
            real, intent(inout) rho_inout,
            integer, intent(in) phase_in,
            real, intent(in) rho_releps,
            real, intent(in) p_releps,
            logical, intent(in), optional meta_extrem,
            real, intent(in), optional prho_init,
            real, intent(in), optional dpdrho_init )
```

A Newton solver targeted at solving for density from a function P = P(rho). fun is a function handle computing pressure derivatives, having the form fun(x, param, p, dpdrho, d2pdrho2) where the last three arguments are optional.

The initial value of rho is the initial guess. It should be an overestimate for liquids, and an underestimate (e.g. 1e-6) for gas. Equals -1 if no solution exists for the given phase. A solution always exists if present(meta_extrem).

Author

Ailo A, November 2014

Parameters

in	*param*	temperature dependent parameters
in	*redt*	reduced temperature
in	*p*	pressure
in	*phase_in*	the phase in which we desire a density.
in	*rho_releps*	relative tolerance for density
in	*p_releps*	relative tolerance for pressure
in	*meta_extrem*	metastable extremum
in	*prho_init*	if one happens to know the value of p when fun is evaluated at rho_inout
in	*dpdrho_init*	if one happens to know the value of dpdrho when fun is evaluated at rho_inout
in,out	*rho_inout*	the computed density; comes in with the initial guess

5.32 meosdatadb Module Reference

Automatically generated file meosdatadb.f90 Time stamp: 2023-03-23T14:18:36.228460.

Data Types

- type meosdata

Variables

- integer, parameter **meos_max_n** = 54
- integer, parameter **meos_max_n_gauss** = 14
- integer, parameter **meos_max_n_nona** = 3
- integer, parameter **meos_id_max_n** = 13
- type(meosdata), parameter **meos_1** = meosdata(ident = "PXYL", name = "pxylene", default_ref_state = "NBP", bibref = "DOI: 10.1063/1.3703506", mw = 106.165, tc = 616.168, pc = 3531.5, rhoc = 2.↩ 69392, ttr = 286.4, ptr = 0.580, t_nbp = 411.470, tr = 616.168, rhor = 2.69392, Rgas = 8.314472, acf = 0.324, t_max = 700.0, p_max = 200000.0, n_poly_eos = 7, n_exp_eos = 5, n_gauss_eos = 4, n↩ _nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.0010786811,-0.103161822,0.0421544125, 1.47865376,-2.4266,-0.46575193, 0.190290995,-1.06376565,-0.209934069, 1.25159879,-0.951328356,-0.0269980032, 1.37103180,-0.494160616,-0.0724317468, -3.69464746,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.83,0.83, 0.281,0.932,1.1, 0.443,2.62,2.5, 1.2,3.0,0.778, 1.13,4.5,2.2, 2.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0,0.0d0 /), d_eos = (/ 5,1,4,1,1,2, 3,1,3,2,2,7, 1,1,3,3,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,2,2,1,2,1, 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -1.179,-1.065,-1.764,-13.675,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -2.445,-1.↩ 483,-4.971,-413.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.↩ 267,0.4242,0.864,1.1465,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos =

(/ 0.54944,0.7234,0.4926,0.8459,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_↩
exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos =
(/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0
/), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /),
big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/ 5.↩
2430504,5.2291378,19.549862, 16.656178,5.9390291,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0
/), t_id = (/ 0.0,414.0,1256.0, 2649.0,6681.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id =
5.9815277224498971, a2_id = -0.5247807538556827)

- type(meosdata), parameter **meos_2** = meosdata(ident = "C2", name = "ethane", default_ref_state = "NBP", bibref = "DOI: 10.1063/1.1859286", mw = 30.06904, tc = 305.322, pc = 4872.2, rhoc = 6.856886685, ttr = 90.368, ptr = 0.001142, t_nbp = 184.569, tr = 305.322, rhor = 6.856886685, Rgas = 8.314472, acf = 0.0995, t_max = 675.0, p_max = 900000.0, n_poly_eos = 5, n_exp_↩
eos = 34, n_gauss_eos = 5, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.83440745735241,-1.↩
4287360607171,0.34430242210927, -0.42096677920265,0.012094500886549,-0.57976201597341, -0.↩
033127037870838,-0.11751654894130,-0.11160957833067, 0.062181592654406,0.098481795434443,-
0.098268582682358, -0.00023977831007049,0.00069885663328821,0.19665987803305e-4, -0.↩
014586152207928,0.046354100536781,0.0060764622180645, -0.0026447330147828,-0.042931872689904,0.↩
0029987786517263, 0.005291933517501,-0.0010383897798198,-0.054260348214694, -0.21959362918493,0.↩
35362456650354,-0.12477390173714, 0.18425693591517,-0.16192256436754,-0.082770876149064, 0.↩
050160758096437,0.0093614326336655,-0.00027839186242864, 0.23560274071481e-4,0.0039238329738527,-
0.00076488325813618, -0.004994430444073,0.0018593386407186,-0.00061404353331199, -0.↩
0023312179367924,0.002930104790876,-0.00026912472842883, 184.13834111814,-10.397127984854,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.0,0.25, 0.75,0.75,2.0,
4.25,0.75,2.25, 3.0,1.0,1.25, 2.75,1.0,2.0, 2.5,5.5,7.0, 0.5,5.5,2.5, 4.0,2.0,10.0, 16.0,18.0,20.0, 14.0,18.↩
0,12.0, 19.0,7.0,15.0, 9.0,26.0,28.0, 28.0,22.0,13.0, 0.0,3.0,3.0, 0.0,3.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,2,2,4,1, 1,2,2,3,6,6, 7,9,10,2,4,4, 5,5,6,8,9,2, 3,3,3,4,4,5,
5,6,11,14,3,3, 4,8,10,1,1,3, 3,2,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,1, 1,1,1,1,1,1, 1,1,1,2,2,2,
2,2,2,2,2,3, 3,3,3,3,3,3, 3,3,3,3,4,4, 4,4,4,2,2,2, 2,2,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0,
1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.↩
0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0,
1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), eta_eos = (/ -15.0,-15.0,-15.0,-20.0,-20.0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0 /), beta_eos = (/ -150.0,-150.0,-150.0,-275.0,-400.0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0 /), gamma_eos = (/ 1.05,1.05,1.05,1.22,1.16,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0 /), epsilon_eos = (/ 1.0,1.0,1.0,1.0,1.0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /),
tau_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos
= (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0
/), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0
/), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/
4.003039265,1.117433359,3.467773215, 6.94194464,5.970850948,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,430.23083,1224.3159, 2014.12064,4268.34363,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -3.0938209725939316, a2_id = 3.2503021100636769)
- type(meosdata), parameter **meos_3** = meosdata(ident = "CO", name = "co", default_ref_state = "NBP", bibref = "DOI: 10.1021/je050186n", mw = 28.0101, tc = 132.86, pc = 3494.0, rhoc = 10.85, ttr = 68.16, ptr = 15.↩
53, t_nbp = 81.64, tr = 132.86, rhor = 10.85, Rgas = 8.314472, acf = 0.0497, t_max = 500.0, p_max = 100000.0, n_poly_eos = 6, n_exp_eos = 6, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.90554,-2.4515,0.53149, 0.024173,0.072156,0.00018818, 0.19405,-0.043268,-0.12778, -0.027896,-0.↩
034154,0.016329, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.125,1.↩
5, 1.375,0.25,0.875, 0.625,1.75,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,
0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,
0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.↩
0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩

0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos =
(/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.↩
0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na
= (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na
= (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 2, n_id = 3, c_id = (/ 3.5,0.22311e-
6,1.0128, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,1.5,3089.0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -3.3728307183251847, a2_id =
3.3683452997752736)

- type(meosdata), parameter **meos_4** = meosdata(ident = "O-H2", name = "orthohyd", default_ref_state =
"IDGAS", bibref = "DOI: 10.1063/1.3160306", mw = 2.01594, tc = 33.22, pc = 1310.65, rhoc = 15.↩
445, ttr = 14.008, ptr = 7.560, t_nbp = 20.38, tr = 33.22, rhor = 15.445, Rgas = 8.314472, acf = -0.↩
218, t_max = 1000.0, p_max = 2000000.0, n_poly_eos = 7, n_exp_eos = 2, n_gauss_eos = 5, n_nona↩
_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.01,-6.83148,2.11505, 4.38353,0.211292,-1.00939, 0.142086,-
0.876960,0.804927, -0.710775,0.0639688,0.0710858, -0.087654,0.647088,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.7333,1.1372, 0.5136,0.5638,1.6248, 1.8290,2.4040,2.1050, 4.↩
1,7.658,1.259, 7.589,3.9460,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,1,2,2,
3,1,3,2,1,3, 1,1,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_↩
eos = (/ 0,0,0,0,0,0, 0,1,1,2,2,2, 2,2,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,
0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -1.169,-0.894,-0.04,-2.072,-
1.306,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -0.4555,-0.4046,-0.0869,-
0.4415,-0.5743,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.5444,0.↩
6627,0.763,0.6587,1.4327,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.↩
6366,0.3876,0.9437,0.3976,0.9626,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_↩
eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/
0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0
/), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.↩
0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c↩
_id = (/ 2.5,2.54151,-2.3661, 1.00365,1.22447,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id
= (/ 0.0,856.0,1444.0, 2194.0,6968.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -5.↩
6016156823632457, a2_id = 5.1318379867780717)
- type(meosdata), parameter **meos_5** = meosdata(ident = "IC5", name = "ipentane", default_ref_state = "NBP",
bibref = "DOI: 10.1021/je050186n", mw = 72.14878, tc = 460.35, pc = 3378.0, rhoc = 3.271, ttr = 112.65, ptr
= 0.00000008952, t_nbp = 300.98, tr = 460.35, rhor = 3.271, Rgas = 8.314472, acf = 0.2274, t_max =
500.0, p_max = 1000000.0, n_poly_eos = 6, n_exp_eos = 6, n_gauss_eos = 0, n_nona_eos = 0, n_assoc↩
_eos = 0, n_eos = (/ 1.0963,-3.0402,1.0317, -0.15410,0.11535,0.00029809, 0.39571,-0.045881,-0.35804,
-0.10107,-0.035484,0.018156, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/
0.25,1.125,1.5, 1.375,0.25,0.875, 0.625,1.75,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,
0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,
0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.↩
0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos

= (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.↩
0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/ 4.0,7.4056,9.5772, 15.765,12.119,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,442.0,1109.0, 2069.↩
0,4193.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 2.5822358374928172, a2_id = 1.↩
1609085086424664)

- type(meosdata), parameter **meos_6** = meosdata(ident = "NC10", name = "decane", default_ref_state = "NBP", bibref = "DOI: 10.1021/je050186n", mw = 142.28168, tc = 617.7, pc = 2103.0, rhoc = 1.64, ttr = 243.5, ptr = 0.001404, t_nbp = 447.27, tr = 617.7, rhor = 1.64, Rgas = 8.314472, acf = 0.4884, t_max = 675.0, p_max = 800000.0, n_poly_eos = 6, n_exp_eos = 6, n_gauss_eos = 0, n_nona_eos = 0, n_assoc↩
_eos = 0, n_eos = (/ 1.0461,-2.4807,0.74372, -0.52579,0.15315,0.00032865, 0.84178,0.055424,-0.73555, -0.18507,-0.020775,0.012335, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.125,1.5, 1.375,0.25,0.875, 0.625,1.75,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.↩
0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.↩
0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/ 19.109,25.685,28.233, 12.417,10.035,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,1193.0,2140.0, 4763.↩
0,10862.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 13.936202857079877, a2_id = -10.5265173263752523)

- type(meosdata), parameter **meos_7** = meosdata(ident = "R21", name = "r21", default_ref_state = "IIR", bibref = "ISBN: 978-3-662-02610-6", mw = 102.92, tc = 451.48, pc = 5181.20, rhoc = 5.1107656, ttr = 142.8, ptr = 0.00006828, t_nbp = 282.01, tr = 451.48, rhor = 5.1107656, Rgas = 8.31451, acf = 0.2061, t_max = 473.0, p_max = 138000.0, n_poly_eos = 16, n_exp_eos = 6, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_↩
eos = 0, n_eos = (/ -44.386484873,9.26505600935,-0.551709104376, 0.504676623431,-0.732431415692,-0.868403860387, 0.146234705555,-0.280576335053,0.864743656093, -2.70767233732,3.30476390706,-0.210878239171, 0.449531449589,0.120779813143,-0.277297953777, 0.0305441291172,44.386484873,-9.26505600935, 0.551709104376,1.21128809552,0.167119476587, -0.0504876793028,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 3.↩
0d0,4.0d0,5.0d0, 0.0d0,1.0d0,2.0d0, 3.0d0,4.0d0,0.0d0, 1.0d0,2.0d0,0.0d0, 1.0d0,0.0d0,1.0d0, 1.0d0,3.↩
0d0,4.0d0, 5.0d0,3.0d0,4.0d0, 5.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 0,0,0,1,1,1, 1,1,2,2,2,3, 3,4,4,5,0,0, 0,2,2,2,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,2,2, 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.↩
0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.07470252,0.07470252, 0.↩
07470252,0.07470252,0.07470252, 0.07470252,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩

0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_↩
exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /),
a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na
= (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na
= (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 5, n_id = 5, c_id = (/ 2.941811386600052,0.015738255695164235,4.↩
01226227210022e-06, -3.085217165653779e-08,2.125621923119943e-11,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,1.0,2.0, 3.0,4.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0
/), a1_id = -13.1877035704659669, a2_id = 9.1813742728328709)

- type(meosdata), parameter **meos_8** = meosdata(ident = "ETOH", name = "ethanol", default_ref_state = "NBP", bibref = "DOI: 10.1063/1.4895394", mw = 46.06844, tc = 514.71, pc = 6268.0, rhoc = 5.93, ttr = 159.0, ptr = 0.0000007184, t_nbp = 351.57, tr = 514.71, rhor = 5.93, Rgas = 8.314472, acf = 0.↩
646, t_max = 650.0, p_max = 280000.0, n_poly_eos = 6, n_exp_eos = 10, n_gauss_eos = 9, n_nona_↩
eos = 0, n_assoc_eos = 0, n_eos = (/ 0.058200796,0.94391227,-0.80941908, 0.55359038,-1.4269032,0.↩
13448717, 0.42671978,-1.1700261,-0.92405872, 0.34891808,-0.91327720,0.022629481, -0.15513423,0.↩
21055146,-0.21997690, -0.0065857238,0.75564749,0.10694110, -0.069533844,-0.24947395,0.027177891,
-0.0009053953,-0.12310953,-0.08977971, -0.39512601,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,1.04,2.72, 1.174,1.329,0.195, 2.43,1.274,4.16, 3.3,4.177,2.↩
5, 0.81,2.02,1.606, 0.86,2.5,3.72, 1.19,3.25,3.0, 2.0,2.0,1.0, 1.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,2,2,3, 1,1,1,3,3,2, 2,6,6,8,1,1, 2,3,3,2,2,2,
1,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,1,2,1, 2,1,1,1,2,2,
2,2,2,2,2,2, 2,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.↩
0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), eta_eos = (/ -1.075,-0.463,-0.876,-1.108,-0.741,-4.032, -2.453,-2.3,-3.143,0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0 /), beta_eos = (/ -1.207,-0.0895,-0.581,-0.947,-2.356,-27.01, -4.542,-1.287,-3.090,0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.194,1.986,1.583,0.756,0.495,1.002, 1.077,1.493,1.542,0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.779,0.805,1.869,0.694,1.312,2.054, 0.441,0.793,0.↩
313,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,2, 2,2,2,0,0,0, 0,0 /), del_exp_eos = (/
2,2,2,2,2,2, 2,2,2,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/
0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.↩
0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/
0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/ 4.43069,2.14326,5.09206, 6.60138,5.70777,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,420.4,1334.0, 1958.0,4420.0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -3.6443990342909416, a2_id = 5.0708070582887474)
- type(meosdata), parameter **meos_9** = meosdata(ident = "KR", name = "krypton", default_ref_state = "NBP", bibref = "DOI: 10.1021/je050186n", mw = 83.798, tc = 209.48, pc = 5525.0, rhoc = 10.85, ttr = 115.775, ptr = 73.53, t_nbp = 119.73, tr = 209.48, rhor = 10.85, Rgas = 8.314472, acf = -0.000894, t_max = 750.0, p_max = 200000.0, n_poly_eos = 6, n_exp_eos = 6, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.83561,-2.3725,0.54567, 0.014361,0.066502,0.0001931, 0.16818,-0.033133,-0.15008, -0.022897,-0.↩
021454,0.0069397, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.125,1.↩
5, 1.375,0.25,0.875, 0.625,1.75,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,
0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,
0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.↩
0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.↩

0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos =
(/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.↩
0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /),
beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /),
big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 1, c_id = (/ 2.↩
5,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -3.7506404605274408, a2_id =
3.7798013718120207)

- type(meosdata), parameter **meos_10** = meosdata(ident = "IC4", name = "isobutan", default_ref_state =
"IIR", bibref = "DOI: 10.1063/1.1901687", mw = 58.1222, tc = 407.81, pc = 3629.0, rhoc = 3.879756788,
ttr = 113.73, ptr = 0.00002289, t_nbp = 261.401, tr = 407.81, rhor = 3.879756788, Rgas = 8.314472, acf
= 0.184, t_max = 575.0, p_max = 200000.0, n_poly_eos = 7, n_exp_eos = 16, n_gauss_eos = 2, n↩
_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 2.0686820727966,-3.6400098615204,0.51968754427244,
0.17745845870123,-0.12361807851599,0.045145314010528, 0.03047647996598,0.75508387706302,-0.↩
85885381015629, 0.036324009830684,-0.01954879945055,-0.004445239290496, 0.004641076366646,-
0.071444097992825,-0.080765060030713, 0.15560460945053,0.0020318752160332,-0.10624883571689,
0.039807690546305,0.016371431292386,0.00053212200682628, -0.0078681561156387,-0.0030981191888963,-
0.042276036810382, -0.0053001044558079,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.5,1.0,1.5, 0.0,0.5,0.5, 0.75,2.0,2.5, 2.5,1.5,1.0, 1.5,4.0,7.0, 3.0,7.0,3.↩
0, 1.0,6.0,0.0, 6.0,13.0,2.0, 0.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,4, 4,1,1,2,7,8, 8,1,2,3,3,4, 5,5,10,2,6,1, 2,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,
0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,1,1,1,1,1, 1,2,2,2,2,2, 2,2,2,3,3,2, 2,0,0,0,0,0, 0,0,0,0,0,0,
0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0,
1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -10.0,-10.↩
0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -150.↩
0,-200.0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/
1.16,1.13,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/
0.85,1.0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/
2,2,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0
/), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na =
(/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na =
(/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/ 4.05956619,4.↩
94641014,4.09475197, 15.6632824,9.73918122,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id
= (/ 0.0,387.94064,973.80782, 1772.71103,4228.52424,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0
/), a1_id = -5.9995196315283579, a2_id = 5.0189508361247883)
- type(meosdata), parameter **meos_11** = meosdata(ident = "C1", name = "methane", default_ref_state =
"NBP", bibref = "DOI: 10.1063/1.555898", mw = 16.0428, tc = 190.564, pc = 4599.2, rhoc = 10.139128,
ttr = 90.6941, ptr = 11.696, t_nbp = 111.667, tr = 190.564, rhor = 10.139128, Rgas = 8.31451, acf =
0.01142, t_max = 625.0, p_max = 1000000.0, n_poly_eos = 13, n_exp_eos = 23, n_gauss_eos = 4, n_↩
nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.04367901028,0.6709236199,-1.765577859, 0.8582330241,-
1.206513052,0.512046722, -0.0004000010791,-0.01247842423,0.03100269701, 0.001754748522,-0.↩
3171921605e-5,-0.224034684e-5, 0.2947056156e-6,0.1830487909,0.1511883679, -0.4289363877,0.↩
06894002446,-0.01408313996, -0.0306305483,-0.02969906708,-0.01932040831, -0.1105739959,0.↩
09952548995,0.008548437825, -0.06150555662,-0.04291792423,-0.0181320729, 0.0344590476,-0.↩
00238591945,-0.01159094939, 0.06641693602,-0.0237154959,-0.03961624905, -0.01387292044,0.↩
03389489599,-0.002927378753, 0.9324799946e-4,-6.287171518,12.71069467, -6.423953466,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ -0.5,0.5,1.0,
0.5,1.0,1.5, 4.5,0.0,1.0, 3.0,1.0,3.0, 3.0,0.0,1.0, 2.0,0.0,0.0, 2.0,2.0,5.0, 5.0,5.0,2.0, 4.0,12.0,8.0, 10.0,10.↩
0,10.0, 14.0,12.0,18.0, 22.0,18.0,14.0, 2.0,0.0,1.0, 2.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,2,2, 2,3,4,4,8,9, 10,1,1,1,2,4, 5,6,1,2,3,4,
4,3,5,5,8,2, 3,4,4,4,5,6, 2,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,1,1,1,1,1,

1,1,2,2,2,2, 2,3,3,3,3,4, 4,4,4,4,4,4, 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), eta_eos = (/ -20.0,-40.0,-40.0,-40.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -200.0,-250.0,-250.0,-250.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.07,1.11,1.11,1.11,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 1.0,1.0,1.0,1.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 6, c_id = (/ 4.0016,0.008449,4.6942, 3.4865,1.6572,1.4115, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,648.0,1957.0, 3895.0,5705.0,15080.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -2.9705496667947529, a2_id = 2.8907453831087553)

- type(meosdata), parameter **meos_12** = meosdata(ident = "NC6", name = "hexane", default_ref_state = "NBP", bibref = "Unpublished. Thol, Lemmon and Span (2019). Fundamental Equations of State for Hy", mw = 86.17536, tc = 507.82, pc = 3044.1, rhoc = 2.706, ttr = 177.83, ptr = 0.001189, t_nbp = 341.866, tr = 507.82, rhor = 2.706, Rgas = 8.3144598, acf = 0.30, t_max = 600.0, p_max = 100000.0, n_poly_eos = 5, n_exp_eos = 5, n_gauss_eos = 5, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.0446249,1.740621,-2.050688, -0.7724346,0.2116422,-3.187864, -1.134719,0.6817086,-1.111314, -0.02377251,3.744662,-0.4322223, -0.8054799,-1.055577,-0.02932639, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.303,0.845, 1.055,0.523,1.48, 2.06,1.19,1.883, 0.98,0.94,1.47, 1.11,0.895,1.73, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,2,3,1, 3,2,2,7,1,3, 2,2,1,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,2, 2,1,2,1,2,2, 2,2,2,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -0.864,-1.094,-0.751,-1.062,-6.6,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -0.7,-0.52,-0.69,-0.88,-180.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.21,1.13,1.0,0.82,1.14,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.767,0.707,0.692,0.46,0.92,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/ 4.0,9.21,6.04, 25.3,10.96,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,190.0,3000.0, 1500.0,4500.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 12.3137913581698513, a2_id = -1.3163412546284243)
- type(meosdata), parameter **meos_13** = meosdata(ident = "N2", name = "nitrogen", default_ref_state = "IDGAS", bibref = "DOI: 10.1063/1.1349047", mw = 28.01348, tc = 126.192, pc = 3395.8, rhoc = 11.1839, ttr = 63.151, ptr = 12.5198, t_nbp = 77.3550, tr = 126.192, rhor = 11.1839, Rgas = 8.31451, acf = 0.0372, t_max = 2000.0, p_max = 2200000.0, n_poly_eos = 6, n_exp_eos = 26, n_gauss_eos = 4, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.924803575275,-0.492448489428,0.661883336938, -1.92902649201,-0.0622469309629,0.349943957581, 0.564857472498,-1.61720005987,-0.481395031883, 0.421150636384,-0.0161962230825,0.172100994165, 0.00735448924933,0.0168077305479,-0.00107626664179, -0.0137318088513,0.000635466899859,0.00304432279419, -0.0435762336045,-0.0723174889316,0.0389644315272, -0.021220136391,0.00408822981509,-0.551990017984e-4, -0.0462016716479,-0.00300311716011,0.0368825891208, -0.0025585684622,0.00896915264558,-0.0044151337035, 0.00133722924858,0.000264832491957,19.6688194015, -20.911560073,0.0167788306989,2627.67566274, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,0.875,0.5, 0.875,0.375,0.75, 0.5,0.75,2.0, 1.25,3.5,1.0, 0.5,3.0,0.0, 2.75,0.75,2.5, 4.0,6.0,6.0, 3.0,3.0,6.0, 16.0,11.0,15.0, 12.0,12.0,7.0, 4.0,16.0,0.0, 1.0,2.0,3.0, 0.0d0,0.0d0,0.0d0,

0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,2,2,3,3, 1,1,1,3,3,4, 6,6,7,7,8,8, 1,2,3,4,5,8, 4,5,5,8,3,5, 6,9,1,1,3,2, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,1,1,1,1, 1,1,1,1,1,1, 2,2,2,2,2,2, 3,3,3,3,4,4, 4,4,2,2,2,2, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.↩
0d0,1.0d0, 1.0d0,1.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -20.0,-20.0,-15.0,-25.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -325.0,-325.0,-300.0,-275.0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.16,1.16,1.13,1.25,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 1.0,1.0,1.0,1.0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 4, n_id = 5, c_id = (/ 3.5,3.066469e-6,4.70124e-9, -3.987984e-13,1.012941,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,1.0,2.0, 3.0,3364.011,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -12.7695270804627707, a2_id = -0.0078416296806012)

- type(meosdata), parameter **meos_14** = meosdata(ident = "NC12", name = "c12", default_ref_state = "NBP", bibref = "DOI: 10.1021/ef0341062", mw = 170.33484, tc = 658.1, pc = 1817.0, rhoc = 1.33, ttr = 263.6, ptr = 0.0006262, t_nbp = 489.442, tr = 658.1, rhor = 1.33, Rgas = 8.314472, acf = 0.574, t_max = 700.↩
0, p_max = 700000.0, n_poly_eos = 6, n_exp_eos = 6, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 1.38031,-2.85352,0.288897, -0.165993,0.0923993,2.82772e-4, 0.956627,0.0353076,-0.↩
445008, -0.118911,-0.0366475,0.0184223, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.32,1.23,1.5, 1.4,0.07,0.8, 2.16,1.1,4.1, 5.6,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.↩
0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.↩
0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/ 23.085,37.776,29.369, 12.461,7.7733,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,1280.0,2399.0, 5700.↩
0,13869.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 20.5642558466316814, a2_id = -15.593070164198167)
- type(meosdata), parameter **meos_15** = meosdata(ident = "D2", name = "d2", default_ref_state = "NBP", bibref = "DOI: 10.1063/1.4864752", mw = 4.0282, tc = 38.34, pc = 1679.6, rhoc = 17.23, ttr = 18.724, ptr = 17.189, t_nbp = 23.661, tr = 38.34, rhor = 17.23, Rgas = 8.3144598, acf = -0.136, t_max = 600.0, p_max = 2000000.0, n_poly_eos = 8, n_exp_eos = 6, n_gauss_eos = 7, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.006267958,10.53609,-10.14149, 0.3560610,0.1824472,-1.129638, -0.0549812,-0.6791329,1.347918, -0.8657582,1.719146,-1.917977, 0.1233365,-0.07936891,1.686617, -4.240326,1.857114,-0.5903705, 1.↩
520171,2.361373,-2.297315, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.462,0.5584, 0.627,1.201,0.309, 1.314,1.1166,1.25, 1.↩
25,1.395,1.627, 1.0,2.5,0.635, 0.664,0.7082,2.25, 1.524,0.67,0.709, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,2,3,1, 3,2,2,2,1,1,

3,2,1,1,2,3, 3,1,3,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,0,1,1,2,2, 2,2,2,2,2,2, 2,2,2,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -0.868,-0.636,-0.668,-0.650,-0.745,-0.782, -0.693,0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -0.613,-0.584,-0.570,-1.056,-1.010,-1.025, -1.029,0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.6306,0.7110,0.6446,0.8226,0.9920,1.2184, 1.2030,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 1.460,1.7864,1.647,0.541,0.969,1.↩
892, 1.076,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,2, 2,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,2,2, 2,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d↩
_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 13, c_id = (/ 2.5,-3.54145,3.0326, -3.52422,-1.73421,-3.↩
57135, 2.14858,6.23107,-3.30425, 6.23098,-3.57137,3.32901, 0.97782 /), t_id = (/ 0.0,7174.1,8635.0, 902.↩
7,181.1,438.5, 5034.2,269.9,229.9, 666.4,452.8,192.0, 1187.6 /), a1_id = -2.0677350466039846, a2_id = 2.4237150686246927)

- type(meosdata), parameter **meos_16** = meosdata(ident = "SO2", name = "so2", default_ref_state = "NBP", bibref = "DOI: 10.1021/acs.jced.6b00195", mw = 64.0638, tc = 430.64, pc = 7886.6, rhoc = 8.078, ttr = 197.↩
7, ptr = 1.6661, t_nbp = 263.137, tr = 430.64, rhor = 8.078, Rgas = 8.3144598, acf = 0.256, t_max = 525.0, p_max = 35000.0, n_poly_eos = 5, n_exp_eos = 5, n_gauss_eos = 6, n_nona_eos = 0, n_assoc_eos = 0, n↩
_eos = (/ 0.01744413,1.814878,-2.246338, -0.4602906,0.1097049,-0.9485769, -0.8751541,0.4228777,-0.↩
4174962, -0.002903451,1.64041,-0.4103535, -0.08316597,-0.2728126,-0.1075782, -0.4348434,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.45,0.9994, 1.0d0,0.45,2.907, 2.992,0.87,3.302, 1.002,1.↩
15,0.997, 1.36,2.086,0.855, 0.785,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,2,3,1, 3,2,2,7,1,1, 3,2,2,1,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,2, 2,1,2,1,2,2, 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -1.061,-0.945,-1.741,-1.139,-1.644,-0.647, 0.0d0,0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -0.967,-2.538,-2.758,-1.062,-1.039,-0.41, 0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.276,0.738,0.71,0.997,1.35,0.919, 0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.832,0.69,0.35,0.961,0.981,0.333, 0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,2, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,2,2, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_↩
na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 2, n_id = 4, c_id = (/ 4.0,0.00007397,1.0875, 1.916,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,1.0,783.0, 1864.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -4.541423325625578, a2_id = 4.4732288061807504)
- type(meosdata), parameter **meos_17** = meosdata(ident = "NC8", name = "octane", default_ref_state = "NBP", bibref = "Unpublished. Beckmueller, Thol, Lemmon and Span (2019). Fundamental Equation of ", mw = 114.229, tc = 568.74, pc = 2483.59, rhoc = 2.031, ttr = 216.37, ptr = 0.0020746, t_nbp = 398.794, tr = 568.74, rhor = 2.031, Rgas = 8.3144598, acf = 0.398, t_max = 730.0, p_max = 1000000.0, n_poly_eos = 5, n_↩
exp_eos = 5, n_gauss_eos = 4, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.042240369,1.4800888,-2.0975357, -0.72303256,0.26084383,-1.6713762, -1.3023632,0.67710461,-1.1644509, -0.030939987,3.↩
1437871,-0.011637891, -0.95649696,-0.36897912,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0d0,0.243,0.856, 1.07,0.52,2.3, 2.55,1.075,2.24, 0.951,0.59,0.917, 1.05,1.634,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,2,3,1, 3,2,2,7,1,1, 3,2,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,

0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,2, 2,1,2,1,2,2, 2,2,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.↩ 0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -0.985,-13.6,-1.03,-1.084,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -1.52,-998.0d0,-1.57,-1.44,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_↩ eos = (/ 1.448,1.08,1.185,1.3,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.989,0.986,0.532,1.16,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.↩ 0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 4, c_id = (/ 4.↩ 0,17.47,33.25, 15.63,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,380.0,1724.0, 3881.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 16.93282505786505, a2_id = -4.06060362648397)

- type(meosdata), parameter **meos_18** = meosdata(ident = "R23", name = "r23", default_ref_state = "IIR", bibref = "DOI: 10.1063/1.1559671", mw = 70.01385, tc = 299.293, pc = 4832.0, rhoc = 7.52, ttr = 118.02, ptr = 0.05804, t_nbp = 191.132, tr = 299.293, rhor = 7.52, Rgas = 8.314472, acf = 0.↩ 263, t_max = 475.0, p_max = 120000.0, n_poly_eos = 5, n_exp_eos = 12, n_gauss_eos = 0, n_↩ nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 7.041529,-8.259512,0.00805304, -0.08617615,0.00633341,-0.1863285, 0.3280510,0.5191023,0.06916144, -0.005045875,-0.01744221,-0.05003972, 0.04729813,-0.↩ 06164031,0.01583585, -0.00179579,-0.001099007,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.744,0.94,4.↩ 3, 1.46,0.68,4.8, 1.5,2.07,0.09, 9.6,0.19,11.2, 0.27,1.6,10.3, 14.0,15.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,5,1, 2,3,5,1,2,2, 4,4,4,2,2,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,1, 1,1,1,2,2,2, 2,2,2,3,4,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.↩ 0d0,1.0d0,1.0d0, 1.0d0,1.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.↩ 0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_↩ na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/ 3.999,2.371,3.↩ 237, 2.61,0.8274,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,744.0,1459.0, 2135.↩ 0,4911.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -8.3138729985195088, a2_id = 6.↩ 5508807867641732)
- type(meosdata), parameter **meos_19** = meosdata(ident = "AR", name = "argon", default_ref_state = "IDGAS", bibref = "DOI: 10.1063/1.556037", mw = 39.948, tc = 150.687, pc = 4863.0, rhoc = 13.↩ 40742965, ttr = 83.8058, ptr = 68.891, t_nbp = 87.302, tr = 150.687, rhor = 13.40742965, Rgas = 8.31451, acf = -0.00219, t_max = 2000.0, p_max = 1000000.0, n_poly_eos = 12, n_exp_↩ eos = 25, n_gauss_eos = 4, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.08872230499,0.↩ 705148051673,-1.68201156541, -0.149090144315,-0.120248046009,-0.121649787986, 0.400359336268,-0.271360626991,0.242119245796, 0.00578895831856,-0.0410973356153,0.0247107615416, -0.↩ 321813917507,0.332300176958,0.0310199862873, -0.0307770860024,0.0938911374196,-0.090643210682, -0.000457783492767,-0.826597290252e-4,0.000130134156031, -0.011397840002,-0.0244551699605,-0.064324067176, 0.0588894710937,-0.00064933552113,-0.0138898621584, 0.404898392969,-0.↩ 386125195947,-0.188171423322, 0.159776475965,0.0539855185139,-0.028953417958, -0.0130254133814,0.↩ 00289486967758,-0.00226471343048, 0.00176164561964,0.00585524544828,-0.6925190827, 1.↩ 53154900305,-0.00273804474498,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩

0d0,0.0d0,0.0d0 /), t_eos = (/ 0.0,0.25,1.0, 2.75,4.0,0.0, 0.25,0.75,2.75, 0.0,2.0,0.75, 3.0,3.5,1.0, 2.0,4.0,3.0, 0.0,0.5,1.0, 1.0,7.0,5.0, 6.0,6.0,10.0, 13.0,14.0,11.0, 14.0,8.0,14.0, 6.0,7.0,24.0, 22.0,3.0,1.0, 0.0,0.0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,1,1,2, 2,2,2,3,3,4, 1,1,3,4,4,5, 7,10,10,2,2,4, 4,8,3,5,5,6, 6,7,7,8,9,5, 6,2,1,2,3,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_↩ eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 1,1,1,1,1,1, 1,1,1,2,2,2, 2,2,3,3,3,3, 3,3,3,3,3,4, 4,2,2,2,2,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.↩ 0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -20.0,-20.0,-20.0,-20.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -250.0,-375.0,-300.0,-225.0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.11,1.14,1.17,1.11,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 1.0,1.0,1.0,1.0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 1, c_id = (/ 2.5,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -10.2938148004759711, a2_id = -0.0003415237881503)

- type(meosdata), parameter **meos_20** = meosdata(ident = "R32", name = "r32", default_ref_state = "IIR", bibref = "DOI: 10.1063/1.556002", mw = 52.024, tc = 351.255, pc = 5782.0, rhoc = 8.1500846, ttr = 136.34, ptr = 0.0480, t_nbp = 221.499, tr = 351.255, rhor = 8.1500846, Rgas = 8.314471, acf = 0.2769, t_max = 435.0, p_max = 70000.0, n_poly_eos = 8, n_exp_eos = 11, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_↩ eos = 0, n_eos = (/ 1.046634,-0.5451165,-0.002448595, -0.04877002,0.03520158,0.00162275, 0.2377225e-4,0.029149,0.003386203, -0.004202444,0.0004782025,-0.005504323, -0.02418396,0.4209034,-0.4616537, -1.200513,-2.59155,-1.400145, 0.8263017,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.0,-0.25, -1.0,2.0,2.↩ 0, 0.75,0.25,18.0, 26.0,-1.0,25.0, 1.75,4.0,5.0, 1.0,1.5,1.0, 0.5,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,2,5,1,1,3, 8,4,4,4,8,3, 5,1,1,3,1,2, 3,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,0,4,3,1,4, 1,2,2,1,1,1, 1,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.↩ 0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_↩ exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/ 4.004486,1.160761,2.645151, 5.794987,1.129475,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,798.0,4185.0, 1806.0,11510.0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -8.2581043885434511, a2_id = 6.3531025573429387)
- type(meosdata), parameter **meos_21** = meosdata(ident = "NC22", name = "c22", default_ref_state = "NBP", bibref = "Unpublished. Romeo and Lemmon (2018).", mw = 310.601, tc = 792.2, pc = 1174.0, rhoc = 0.↩ 723, ttr = 317.04, ptr = 0.000003913, t_nbp = 641.298, tr = 792.2, rhor = 0.723, Rgas = 8.3144598, acf = 0.978, t_max = 1000., p_max = 500000., n_poly_eos = 5, n_exp_eos = 5, n_gauss_eos = 5, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.04239455,2.370432,-4.30263, -0.4039603,0.4005704,-2.643419, -0.↩ 9199641,0.1394402,-1.448862, -0.0547678,4.579069,-0.3534636, -0.8217892,-0.2604273,-0.7618884, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.224,0.91, 0.95,0.555,2.36, 3.58,0.5,1.72, 1.↩ 078,1.14,2.43, 1.75,1.1,1.08, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩

0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,2,3,1, 3,2,2,7,1,1, 3,2,2,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,2, 2,1,2,1,2,2, 2,2,2,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -0.641,-1.008,-1.026,-1.21,-0.93,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -0.516,-0.669,-0.25,-1.33,-2.1,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.335,1.187,1.39,1.23,0.763,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.75,1.616,0.47,1.306,0.46,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.↩
0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 3, c_id = (/ 33.9,61.6,77.7, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,1000.0,2400.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 66.7339510994363536, a2_id = -44.1656208449909968)

- type(meosdata), parameter **meos_22** = meosdata(ident = "ALLENE", name = "propadiene", default_ref↩
_state = "NBP", bibref = "Unpublished. Gao, Wu and Lemmon (2017).", mw = 40.06386, tc = 398.0, pc = 5215.6, rhoc = 5.9, ttr = 136.65, ptr = 0.018343, t_nbp = 240.874, tr = 398.0, rhor = 5.9, Rgas = 8.↩
3144598, acf = 0.115, t_max = 400.0, p_max = 10000.0, n_poly_eos = 5, n_exp_eos = 5, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.7231448,-1.790058,-0.06836828, 0.07947672,0.↩
000040778,0.1760558, 0.1443484,-0.1494723,0.008248376, -0.009386559,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.125,1.125,1.25, 0.25,0.75,0.625, 2.0,4.125,4.125, 17.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,2,3,8,2, 3,1,4,3,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,1, 1,2,2,3,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_↩
exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 4, c_id = (/ 4.0,2.1260,7.4934, 5.2240,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,512.0,1442.0, 3896.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -2.7836228624391168, a2_id = 3.2416044956386374)
- type(meosdata), parameter **meos_23** = meosdata(ident = "R1234YF", name = "r1234yf", default_ref_state = "IIR", bibref = "DOI: 10.1021/je200369m", mw = 114.0415928, tc = 367.85, pc = 3382.2, rhoc = 4.↩
17, ttr = 122.77, ptr = 0.000738, t_nbp = 243.665, tr = 367.85, rhor = 4.17, Rgas = 8.314472, acf = 0.↩
276, t_max = 410.0, p_max = 30000.0, n_poly_eos = 5, n_exp_eos = 5, n_gauss_eos = 5, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.04592563,1.546958,-2.355237, -0.4827835,0.1758022,-1.210006, -0.↩
6177084,0.6805262,-0.6968555, -0.02695779,1.389966,-0.4777136, -0.1975184,-1.147646,0.0003428541, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.32,0.929, 0.94,0.38,2.28, 1.76,0.97,2.44, 1.05,1.↩
4,3.0, 3.5,1.0,3.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,2,3,1, 3,2,2,7,1,1, 3,3,2,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,2, 2,1,2,1,2,2,

2,2,2,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.↩
0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -1.02,-1.336,-1.055,-5.84,-16.2,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0 /), beta_eos = (/ -1.42,-2.31,-0.89,-80.0,-108.0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.13,0.67,0.46,1.28,1.2,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0 /), epsilon_eos = (/ 0.712,0.910,0.677,0.718,1.64,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0
/), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na
= (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_↩
na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id
= 1, n_id = 5, c_id = (/ 5.944,7.549,1.537, 2.03,7.455,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0
/), t_id = (/ 0.0,718.0,877.0, 4465.0,1755.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id =
-12.8379485300797782, a2_id = 8.0426156173728476)

- type(meosdata), parameter **meos_24** = meosdata(ident = "P-H2", name = "parahyd", default_ref_state =
"NBP", bibref = "DOI: 10.1063/1.3160306", mw = 2.01588, tc = 32.938, pc = 1285.8, rhoc = 15.538,
ttr = 13.8033, ptr = 7.041, t_nbp = 20.271, tr = 32.938, rhor = 15.538, Rgas = 8.314472, acf = -0.↩
219, t_max = 1000.0, p_max = 2000000.0, n_poly_eos = 7, n_exp_eos = 2, n_gauss_eos = 5, n_nona↩
_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.01,-7.33375,2.60375, 4.66279,0.682390,-1.47078, 0.135801,-
1.05327,0.328239, -0.0577833,0.0449743,0.0703464, -0.0401766,0.119510,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.6855,1.0, 0.489,0.774,1.133, 1.386,1.619,1.162, 3.96,5.276,0.99,
6.791,3.190,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,1,2,2, 3,1,3,2,1,3, 1,1,0,0,0,0,
0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,1,1,2,2,2,
2,2,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.↩
0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -1.7437,-0.5516,-0.0634,-2.1341,-1.7770,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -0.194,-0.2019,-0.0301,-0.2383,-0.3253,0.0d0, 0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.8048,1.5248,0.6648,0.6832,1.4930,0.↩
0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 1.5487,0.1785,1.28,0.6319,1.↩
7104,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0,
0,0 /), del_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.↩
0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0
/), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /),
big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 8, c_id = (/ 2.5,4.30256,13.0289, -47.7365,50.0013,-
18.6261, 0.993973,0.536078,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,499.0,826.5, 970.8,1166.↩
2,1341.4, 5395.0,10185.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -1.4485885457134948, a2_id = 1.↩
8845208741487571)
- type(meosdata), parameter **meos_25** = meosdata(ident = "ACETYLENE", name = "acetylene", default↩
_ref_state = "IIR", bibref = "Unpublished. Gao, Wu and Lemmon (2017).", mw = 26.03728, tc = 308.3,
pc = 5988.2, rhoc = 8.83, ttr = 191.75, ptr = 123.523, t_nbp = 188.260, tr = 308.3, rhor = 8.83, Rgas =
8.3144598, acf = 0.178, t_max = 310.0, p_max = 10000.0, n_poly_eos = 5, n_exp_eos = 5, n_gauss↩
_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.8157856,-1.85265,-0.115457, 0.0938171,0.↩
00006405,0.2031037, 0.1417312,-0.1641216,0.01495196, -0.014536,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.125,1.125,1.25, 0.25,0.75,0.625, 2.0,4.125,4.125,
17.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,2,3,8,2,
3,1,4,3,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/
0,0,0,0,0,1, 1,2,2,3,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0

/), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_↩
exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /),
a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na
= (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na
= (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 4, c_id = (/ 3.5,1.8374,3.4338, 2.6089,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,610.0,1428.0, 6857.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -3.4118831161847254, a2_id = 3.3339613727891444)

- type(meosdata), parameter **meos_26** = meosdata(ident = "R1234ZE", name = "r1234zee", default_ref↩
_state = "IIR", bibref = "DOI: 10.1007/s10765-016-2040-6", mw = 114.0416, tc = 382.513, pc = 3634.↩
9, rhoc = 4.29, ttr = 169.0, ptr = 0.2286, t_nbp = 254.177, tr = 382.513, rhor = 4.29, Rgas = 8.3144598,
acf = 0.313, t_max = 420.0, p_max = 100000.0, n_poly_eos = 5, n_exp_eos = 5, n_gauss_eos = 6, n_↩
nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.03982797,1.812227,-2.537512, -0.5333254,0.1677031,-1.↩
323801, -0.6694654,0.8072718,-0.7740229, -0.01843846,1.407916,-0.4237082, -0.2270068,-0.805213,0.↩
00994318, -0.008798793,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.223,0.755, 1.24,0.44,2.0,
2.2,1.2,1.5, 0.9,1.33,1.75, 2.11,1.0,1.5, 1.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,2,3,1,
3,2,2,7,1,1, 3,3,2,1,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/
0,0,0,0,0,2, 2,1,2,1,2,2, 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /),
g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -1.0,-1.61,-1.24,-9.34,-5.78,-3.08, 0.0d0,0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -1.21,-1.37,-0.98,-171.0d0,-47.4,-15.4, 0.0d0,0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.943,0.642,0.59,1.2,1.33,0.64, 0.0d0,0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.728,0.87,0.855,0.79,1.3,0.71, 0.0d0,0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,2, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/
2,2,2,2,2,2, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/
0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.↩
0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/
0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 3, c_id = (/ 4.0,9.3575,10.717, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,513.0,1972.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0 /), a1_id = -12.5583513312376738, a2_id = 8.7912317462661171)
- type(meosdata), parameter **meos_27** = meosdata(ident = "H2S", name = "h2s", default_ref_state = "NBP",
bibref = "DOI: 10.1021/je050186n", mw = 34.08088, tc = 373.1, pc = 9000.0, rhoc = 10.19, ttr = 187.7, ptr
= 23.25, t_nbp = 212.85, tr = 373.1, rhor = 10.19, Rgas = 8.314472, acf = 0.1005, t_max = 760.0, p_max =
170000.0, n_poly_eos = 6, n_exp_eos = 6, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/
0.87641,-2.0367,0.21634, -0.050199,0.066994,0.00019076, 0.20227,-0.0045348,-0.22230, -0.034714,-0.↩
014885,0.0074154, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.125,1.↩
5, 1.375,0.25,0.875, 0.625,1.75,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,
0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,
0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.↩
0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩

0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/
0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0
/), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na =
(/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/
0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 2, n_id = 4, c_id = (/ 4.0,0.0000014327,1.↩
1364, 1.9721,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,1.5,1823.0, 3965.↩
0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -4.0740759852320352, a2_id = 3.↩
7632130988924168)

- type(meosdata), parameter **meos_28** = meosdata(ident = "NE", name = "neon", default_ref_state = "NBP", bibref = "Unpublished. Thol, Beckmueller, Weiss, Harvey, Lemmon, Jacobsen and Span (2019).", mw = 20.179, tc = 44.4, pc = 2661.63, rhoc = 24.1, ttr = 24.5561, ptr = 43.355, t_nbp = 27.1000, tr = 44.4, rhor = 24.1, Rgas = 8.3144598, acf = -0.0355, t_max = 725.0, p_max = 1000000.0, n_poly_eos = 5, n_exp_eos = 6, n_gauss_eos = 6, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.031522418,3.7716418,-4.27399448, -0.756466758,0.066679921,-0.356928434, -0.053124761,0.9407234,-0.969302374, 0.461243234,0.↩
008184422,7.84771604, 6.39604094,-1.27480579,-5.51887999, -8.76652276,-0.351732709,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0d0,0.431,0.592, 1.105,0.49,2.3, 3.18,1.36,2.0d0, 0.5,1.12,0.41, 0.64,0.579,0.6, 0.52,0.655,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,2,3,1, 3,2,2,4,7,1, 1,3,2,1,2,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,2, 2,1,2,1,1,2, 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.↩
0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -0.76,-2.126,-2.168,-2.033,-0.743,-4.38, 0.0d0,0.0d0,0.0d0,0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -0.537,-0.765,-0.883,-0.751,-0.531,-11.4, 0.0d0,0.0d0,0.0d0,0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.997,1.782,1.663,1.837,1.953,1.658, 0.0d0,0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.5775,0.9137,0.7895,0.6229,0.4992,0.869, 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,2, 0,0,0,0,0,0, 0,0 /), del_↩
exp_eos = (/ 2,2,2,2,2,2, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /),
a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na
= (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na
= (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 1, c_id = (/ 2.5,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -3.0384719151147275, a2_id = 3.253690479855404)
- type(meosdata), parameter **meos_29** = meosdata(ident = "R116", name = "r116", default_ref_state = "IIR", bibref = "DOI: 10.1021/je050186n", mw = 138.01182, tc = 293.03, pc = 3048.0, rhoc = 4.444, ttr = 173.1, ptr = 26.08, t_nbp = 195.06, tr = 293.03, rhor = 4.444, Rgas = 8.314472, acf = 0.2566, t_max = 425.0, p_max = 50000.0, n_poly_eos = 6, n_exp_eos = 6, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 1.1632,-2.8123,0.77202, -0.14331,0.10227,0.00024629, 0.30893,-0.028499,-0.30343, -0.068793,-0.↩
027218,0.010665, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.125,1.↩
5, 1.375,0.25,0.875, 0.625,1.75,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.↩
0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.↩

0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos =
(/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.↩
0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /),
beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /),
big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 4, c_id = (/ 4.0,2.↩
4818,7.0622, 7.9951,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,190.0,622.0,
1470.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -10.7088893891341712, a2_id
= 8.9149145291965084)

- type(meosdata), parameter **meos_30** = meosdata(ident = "CYCLOHEX", name = "cyclohex", default_ref↩
_state = "NBP", bibref = "DOI: 10.1063/1.4900538", mw = 84.15948, tc = 553.6, pc = 4080.5, rhoc = 3.↩
224, ttr = 279.86, ptr = 5.3487, t_nbp = 353.865, tr = 553.6, rhor = 3.224, Rgas = 8.3144598, acf = 0.↩
2096, t_max = 700.0, p_max = 250000., n_poly_eos = 5, n_exp_eos = 5, n_gauss_eos = 10, n_nona_eos =
0, n_assoc_eos = 0, n_eos = (/ 0.05483581,1.607734,-2.375928, -0.5137709,0.1858417,-0.9007515, -0.↩
5628776,0.2903717,-0.3279141, -0.03177644,0.8668676,-0.1962725, -0.1425992,0.004197016,0.1776584,
-0.04433903,-0.03861246,0.07399692, 0.02036006,0.00272825,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.37,0.79, 1.↩
075,0.37,2.4, 2.5,0.5,3.0d0, 1.06,1.6,0.37, 1.33,2.5,0.9, 0.5,0.73,0.2, 1.5,1.5,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/
4,1,1,2,3,1, 3,2,2,7,1,1, 3,3,2,2,3,2, 3,2,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0
/), l_eos = (/ 0,0,0,0,0,2, 2,1,2,1,2,2, 2,2,2,2,2,2, 2,2,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,
0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -0.99,-1.43,-0.97,-1.93,-0.92,-1.27,
-0.87,-0.82,-1.40,-3.0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -0.38,-4.2,-1.2,-0.9,-1.2,-2.6, -5.3,-4.4,-4.↩
2,-25.0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.65,0.63,1.14,0.09,0.56,0.40, 1.01,0.45,0.85,0.86,0.↩
0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.73,0.75,0.48,2.32,0.20,1.33, 0.68,1.11,1.47,0.99,0.0d0,0.0d0,
0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,2, 2,2,2,2,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,2,2, 2,2,2,2,0,0, 0,0
/), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na =
(/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na =
(/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1,
n_id = 5, c_id = (/ 4.0,0.83775,16.036, 24.636,7.1715,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0
/), t_id = (/ 0.0,773.0,941.0, 2185.0,4495.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id =
0.9891146198409615, a2_id = 1.6359656987149183)
- type(meosdata), parameter **meos_31** = meosdata(ident = "R114", name = "r114", default_ref_state =
"IIR", bibref = "ISBN: 978-3-662-02610-6", mw = 170.921, tc = 418.83, pc = 3257.0, rhoc = 3.3932,
ttr = 180.63, ptr = 0.2021, t_nbp = 276.741, tr = 418.83, rhor = 3.3932, Rgas = 8.31451, acf = 0.↩
2523, t_max = 507.0, p_max = 21000.0, n_poly_eos = 16, n_exp_eos = 6, n_gauss_eos = 0, n_↩
nona_eos = 0, n_assoc_eos = 0, n_eos = (/ -0.340776521414,0.32300139842,-0.0424950537596, 1.↩
0793887971,-1.99243619673,-0.155135133506, -0.121465790553,-0.0165038582393,-0.186915808643,
0.308074612567,0.115861416115,0.0276358316589, 0.108043243088,0.0460683793064,-0.174821616881,
0.0317530854287,0.340776521414,-0.32300139842, 0.0424950537596,-1.66940100976,4.08693082002,
-2.41738963889,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0 /), t_eos = (/ 3.0d0,4.0d0,5.0d0, 0.0d0,1.0d0,2.0d0, 3.0d0,4.0d0,0.0d0, 1.0d0,2.0d0,0.↩
0d0, 1.0d0,0.0d0,1.0d0, 1.0d0,3.0d0,4.0d0, 5.0d0,3.0d0,4.0d0, 5.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 0,0,0,1,1,1, 1,1,2,2,2,3,
3,4,4,5,0,0, 0,2,2,2,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0,
0,0,0,0,0,0, 0,0,0,0,2,2, 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_↩
eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0,
1.0d0,1.21103865,1.21103865, 1.21103865,1.21103865,1.21103865, 1.21103865,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,

0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos =
(/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.↩
0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na
= (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/
0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 5, n_id = 5, c_id = (/ 2.00752063361521,0.↩
06662573870618954,-0.00012119710825051628, 1.385185902680976e-07,-7.290628052885859e-11,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,1.0,2.0, 3.0,4.0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -11.3866923427291784, a2_id = 10.9321087031209583)

- type(meosdata), parameter **meos_32** = meosdata(ident = "O2", name = "oxygen", default_ref_state = "IDGAS", bibref = "DOI: 10.1016/0378-3812(85)87016-3", mw = 31.9988, tc = 154.581, pc = 5043.0, rhoc = 13.63, ttr = 54.361, ptr = 0.14628, t_nbp = 90.1878, tr = 154.581, rhor = 13.63, Rgas = 8.↩
31434, acf = 0.0222, t_max = 2000.0, p_max = 82000.0, n_poly_eos = 13, n_exp_eos = 19, n_gauss↩
_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.3983768749,-1.846157454,0.4183473197,
0.02370620711,0.09771730573,0.03017891294, 0.02273353212,0.01357254086,-0.04052698943, 0.↩
0005454628515,0.0005113182277,0.2953466883e-6, -0.8687645072e-4,-0.2127082589,0.08735941958,
0.127550919,-0.09067701064,-0.03540084206, -0.03623278059,0.0132769929,-0.0003254111865, -0.↩
008313582932,0.002124570559,-0.0008325206232, -0.2626173276e-4,0.002599581482,0.009984649663,
0.002199923153,-0.02591350486,-0.1259630848, 0.1478355637,-0.01011251078,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0 /), t_eos = (/ 0.0,1.5,2.5, -0.5,1.5,2.0, 0.0,1.0,2.5, 0.0,2.0,5.0, 2.0,5.0,6.0, 3.5,5.5,3.0,
7.0,6.0,8.5, 4.0,6.5,5.5, 22.0,11.0,18.0, 11.0,23.0,17.0, 18.0,23.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0 /), d_eos = (/ 1,1,1,2,2,2, 3,3,3,6,7,7, 8,1,1,2,2,3, 3,5,6,7,8,10, 2,3,3,4,4,5, 5,5,0,0,0,0, 0,0,0,0,0,0,
0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,2,2,2,2,2, 2,2,2,2,2,2, 4,4,4,4,4,4, 4,4,0,0,0,0,
0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0,
1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.↩
0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos =
(/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos
= (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp↩
_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos
= (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.↩
0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.↩
0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1,
n_id = 6, c_id = (/ 3.51808732,1.02323928,0.784357918, 0.00337183363,-0.0170864084,0.0463751562,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,2246.3244,11259.9763, 1201.26209,69.↩
0089445,5328.05445, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -14.7168366664971053,
a2_id = -0.0110839854084588)
- type(meosdata), parameter **meos_33** = meosdata(ident = "H2", name = "hydrogen", default_ref_state = "NBP", bibref = "DOI: 10.1063/1.3160306", mw = 2.01588, tc = 33.145, pc = 1296.4, rhoc = 15.508, ttr = 13.957, ptr = 7.3578, t_nbp = 20.369, tr = 33.145, rhor = 15.508, Rgas = 8.314472, acf = -0.219, t_max = 1000.0, p_max = 2000000.0, n_poly_eos = 7, n_exp_eos = 2, n_gauss_eos = 5, n_nona_eos = 0, n_assoc↩
_eos = 0, n_eos = (/ 0.01,-6.93643,2.1101, 4.52059,0.732564,-1.34086, 0.130985,-0.777414,0.351944, -
0.0211716,0.0226312,0.032187, -0.0231752,0.0557346,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0 /), t_eos = (/ 1.0,0.6844,0.989, 0.489,0.803,1.1444, 1.409,1.754,1.311, 4.187,5.646,0.791, 7.249,2.↩
986,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,1,2,2, 3,1,3,2,1,3, 1,1,0,0,0,0,
0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,1,1,2,2,2,
2,2,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.↩

0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -1.685,-0.489,-0.103,-2.506,-1.607,0.0d0, 0.0d0,0.0d0,0.0d0,0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -0.171,-0.2245,-0.1304,-0.2785,-0.3967,0.0d0, 0.0d0,0.↩ 0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.7164,1.3444,1.4517,0.7204,1.5445,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 1.506,0.156,1.736,0.67,1.662,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), del_↩ exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 6, c_id = (/ 2.5,1.616,-0.4117, -0.792,0.758,1.217, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,531.0,751.0, 1989.0,2484.0,6859.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -1.4579850777108838, a2_id = 1.8880764164897215)

- type(meosdata), parameter **meos_34** = meosdata(ident = "R124", name = "r124", default_ref_state = "IIR", bibref = "de Vries et al. (1995). Thermodynamic Properties of HCFC 124.", mw = 136.475, tc = 395.425, pc = 3624.295, rhoc = 4.1033156, ttr = 74., ptr = 0.0000000000003228, t_nbp = 261.187, tr = 395.425, rhor = 4.1033156, Rgas = 8.314471, acf = 0.28810, t_max = 470.0, p_max = 40000.0, n_poly_eos = 9, n_exp_eos = 11, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ -0.01262962,2.168373,-3.330033, 0.↩ 1610361,-0.9666145e-4,0.0119131, -0.002880217,0.001681346,0.1594968e-4, 0.1289674,0.1182213e-4,-0.4713997, -0.2412873,0.6868066,-0.08621095, 0.4728645e-5,0.01487933,-0.03001338, 0.001849606,0.↩ 0004126073,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 2.0,0.5,1.0, 0.5,2.5,-1.0, 1.0,0.0,-0.5, 1.5,1.0,2.5, -0.25,1.0,5.0, 2.0,15.0,20.↩ 0, 15.0,45.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,2,3, 5,6,8,2,12,1, 1,1,1,15,3,3, 4,9,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,0,0,1,1,2, 2,2,2,2,3,3, 4,4,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.↩ 0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.↩ 0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.↩ 0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 4, n_id = 4, c_id = (/ 3.175638,0.037376164885882274,-3.3525629318311684e-05, 2.1642904553909864e-08,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,1.0,2.0, 3.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -11.6694278205254687, a2_id = 9.8760554938431273)
- type(meosdata), parameter **meos_35** = meosdata(ident = "MEOH", name = "methanol", default_ref_state = "NBP", bibref = "ISBN: 0-632-03455-6", mw = 32.04216, tc = 513.38, pc = 8215.85, rhoc = 8.78517, ttr = 175.61, ptr = 0.00018629, t_nbp = 337.632, tr = 513.38, rhor = 8.78517, Rgas = 8.31448, acf = 0.5625, t_max = 620.0, p_max = 800000.0, n_poly_eos = 17, n_exp_eos = 19, n_gauss_eos = 8, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ -2.80062505988,12.5636372418,-13.0310563173, 3.2659313406,-4.↩ 11425343805,3.46397741254, -0.083644396759,-0.369240098923,0.00313180842152, 0.603201474111,-0.231158593638,0.106114844945, -0.0792228164995,-0.422419150975e-4,0.00758196739214, -0.↩ 244617434701e-4,0.115080328802e-5,-12.5099747447, 27.0392835391,-21.2070717086,6.3279947227, 14.3687921636,-28.7450766617,18.5397216068, -3.88720372879,-4.16602487963,5.29665875982, 0.↩ 509360272812,-3.30257604839,-0.311045210826, 0.273460830583,0.518916583979,-0.00227570803104, 0.0211658196182,-0.0114335123221,0.00249860798459, -8.19291988442,0.478601004557,-0.444161392885, 0.17962181041,-0.687602278259,2.40459848295, -6.88463987466,1.13992982501,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.0d0,1.0d0,2.0d0, 3.0d0,1.0d0,2.0d0, 3.0d0,4.0d0,6.0d0, 0.0d0,3.0d0,4.0d0, 0.0d0,7.0d0,1.0d0, 6.0d0,7.0d0,1.0d0, 2.0d0,3.0d0,4.0d0, 1.↩ 0d0,2.0d0,3.0d0, 5.0d0,1.0d0,2.0d0, 1.0d0,2.0d0,4.0d0, 5.0d0,2.0d0,5.0d0, 9.0d0,14.0d0,19.0d0, 0.↩

0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,1,2,2, 2,2,2,3,3,3, 4,4,5,6,7,1, 1,1,1,2,2,2, 2,3,4,5,5,5, 5,6,9,6,6,4, 1,1,1,1,1,3, 3,3,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,2, 2,2,2,2,2,2, 2,2,2,2,2,2, 2,2,2,4,4,6, 2,3,2,4,2,3, 2,4,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.↩
0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.01733510223052, 1.↩
01733510223052,1.01733510223052,1.01733510223052, 1.01733510223052,1.01733510223052,1.↩
01733510223052, 1.01733510223052,1.01733510223052,1.01733510223052, 1.01733510223052,1.↩
01733510223052,1.01733510223052, 1.01733510223052,1.01733510223052,1.01733510223052, 1.↩
03497071023039,1.03497071023039,1.05291203329783, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -4.↩
06934040892209,-8.20892015621185,-9.15601592007471,-83.8326275286616,-16.2773616356884,-27.↩
705105527215, -16.2773616356884,-264.95250181898,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 3.8940745646517,3.8940745646517,3.8940745646517,3.8940745646517,3.8940745646517,23.↩
0649031906293, 23.0649031906293,23.0649031906293,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.54080254509371,1.54080254509371,1.54080254509371,1.54080254509371,1.↩
54080254509371,1.08389789427588, 1.08389789427588,1.08389789427588,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 1,1,1,1,1,1, 1,1,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,3,2,4,2,3, 2,4,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.↩
0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 8, c_id = (/ 3.9007912,10.992677,18.33683, -16.366004,-6.2332348,2.8035363, 1.0778099,0.96965697,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,2115.01542,1676.18569, 1935.16717,1504.97016,4222.83691, 5296.17127,273.36934,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -4.4231521797113365, a2_id = 5.↩
4815623918310443)

- type(meosdata), parameter **meos_36** = meosdata(ident = "R12", name = "r12", default_ref_state = "IIR", bibref = "Marx et al. (1992). Neue Zustandsgleichungen fuer R 12, R 22, R 11 und R 113.", mw = 120.913, tc = 385.12, pc = 4136.1, rhoc = 4.672781, ttr = 116.099, ptr = 0.0002425, t_nbp = 243.398, tr = 385.12, rhor = 4.672781, Rgas = 8.314471, acf = 0.17948, t_max = 525.0, p_max = 200000.0, n_poly_eos = 7, n_exp_eos = 15, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 2.075343402,-2.962525996,0.↩
01001589616, 0.01781347612,0.02556929157,0.002352142637, -0.8495553314e-4,-0.01535945599,-0.2108816776, -0.01654228806,-0.0118131613,-0.416029583e-4, 0.2784861664e-4,0.1618686433e-5,-0.1064614686, 0.0009369665207,0.02590095447,-0.04347025025, 0.1012308449,-0.1100003438,-0.↩
003361012009, 0.0003789190008,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.5,1.0,2.0, 2.5,-0.5,0.0, 0.0,-0.5,1.5, 2.5,-0.5,0.0, 0.5,-0.5,4.0, 4.0,2.0,4.0, 12.0,14.0,0.0, 14.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,4,6, 8,1,1,5,7,12, 12,14,1,9,1,1, 3,3,5,9,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,1,1,1,1,1, 1,1,2,2,3,3, 3,3,3,4,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.↩
0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big↩
_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/ 4.00363901052655,3.1606387751370404,0.37125992205225483, 3.↩
5622775274430247,2.1215335661615504,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,1433.4342,2430.0498, 685.65952,412.41579,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -14.7178947415560639, a2_id = 9.4030125798124899)
- type(meosdata), parameter **meos_37** = meosdata(ident = "NC16", name = "c16", default_ref_state = "NBP", bibref = "Unpublished. Romeo and Lemmon (2018).", mw = 226.441, tc = 722.1, pc = 1479.85, rhoc = 1.0, ttr

= 291.329, ptr = 0.00009387, t_nbp = 559.903, tr = 722.1, rhor = 1.0, Rgas = 8.3144598, acf = 0.749, t_max = 800.0, p_max = 50000.0, n_poly_eos = 5, n_exp_eos = 5, n_gauss_eos = 5, n_nona_eos = 0, n_assoc↩ _eos = 0, n_eos = (/ 0.03965879,1.945813,-3.738575, -0.3428167,0.3427022,-2.519592, -0.8948857,0.↩ 10760773,-1.297826, -0.04832312,4.245522,-0.31527585, -0.7212941,-0.2680657,-0.7859567, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.224,0.91, 0.95,0.555,2.36, 3.58,0.5,1.72, 1.078,1.14,2.↩ 43, 1.75,1.1,1.08, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,2,3,1, 3,2,2,7,1,1, 3,2,2,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,2, 2,1,2,1,2,2, 2,2,2,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.↩ 0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -0.641,-1.008,-1.026,-1.21,-0.93,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -0.516,-0.669,-0.25,-1.33,-2.1,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.335,1.187,1.39,1.23,0.763,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.75,1.616,0.47,1.306,0.46,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.↩ 0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 3, c_id = (/ 23.03,18.91,76.23, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,420.0,1860.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 45.9620674049700142, a2_id = -26.1883393966868319)

- type(meosdata), parameter **meos_38** = meosdata(ident = "F6S", name = "sf6", default_ref_state = "IIR", bibref = "DOI: 10.1063/1.3037344", mw = 146.0554192, tc = 318.7232, pc = 3754.983, rhoc = 5.0823174112, ttr = 223.555, ptr = 231.429, t_nbp = 204.9, tr = 318.7232, rhor = 5.0823174112, Rgas = 8.314472, acf = 0.218, t_max = 625.0, p_max = 150000.0, n_poly_eos = 10, n_exp_eos = 12, n_gauss_eos = 14, n_nona↩ _eos = 0, n_assoc_eos = 0, n_eos = (/ 0.54958259132835,-0.87905033269396,-0.84656969731452, 0.↩ 27692381593529,-4.9864958372345,4.8879127058055, 0.036917081634281,0.00037030130305087,0.↩ 039389132911585, 0.00042477413690006,-0.02415001386389,0.059447650642255, -0.38302880142267,0.↩ 32606800951983,-0.029955940562031, -0.086579186671173,4.1600684707562,-4.1398128855814, -0.↩ 55842159922714,0.56531382776891,0.0082612463415545, -0.01020099533808,-0.021662523861406,0.↩ 034650943893908, -0.028694281385812,0.0084007238998053,-0.26969359922498, 9.0415215646344,- 3.7233103557977,-2752.4670823704, 5771.1861697319,-3023.4003119748,2225277.843536, -2305606.↩ 5559032,6391885.2944475,-6079209.1415592, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.125,1.25,1.875, 0.125,1.5,1.↩ 625, 1.5,5.625,0.625, 0.25,6.0,0.25, 4.75,5.375,5.875, 2.0,5.875,6.0, 5.625,5.75,0.0, 0.5,4.0,1.0, 3.0,2.↩ 0,4.0, 3.0,4.0,1.0, 2.0,3.0,3.0, 4.0,3.0,4.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,2,2, 3,3,4,6,1,2, 2,2,3,6,2,2, 4,4,2,2,1,3, 4,1,1,4,3,4, 4,4,1,1,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,0,0,0,1,1, 1,1,1,1,2,2, 2,2,3,3,2,2, 2,2,2,2,2,2, 2,2,2,2,2,2, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.↩ 0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -10.0,-10.0,-10.0,-10.0,-11.0,-25.0, -30.0,-30.0,-30.0,-30.0,-30.0,-30.0, -30.0,-30.0 /), beta_eos = (/ -150.0,-150.0,-150.0,-150.0,-225.0,-300.0, -350.0,-350.0,-350.0,-350.0,-400.↩ 0,-400.0, -400.0,-400.0 /), gamma_eos = (/ 1.13,1.13,1.13,1.16,1.19,1.19, 1.16,1.16,1.16,1.16,1.22,1.22, 1.22,1.22 /), epsilon_eos = (/ 0.85,0.85,0.85,0.85,1.0,1.0, 1.0,1.0,1.0,1.0,1.0,1.0, 1.0,1.0 /), tau_exp_eos = (/ 2,2,2,2,2,2, 2,2,2,2,2,2, 2,2 /), del_exp_eos = (/ 2,2,2,2,2,2, 2,2,2,2,2,2, 2,2 /), b_assoc_eos = (/ 0.↩ 0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 4, c_id = (/ 4.0,3.66118232,7.87885103, 3.45981679,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,515.4653151,875.5693279, 1349.12572,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -15.4089857348096295, a2_id = 10.8151609115241314)

- type(meosdata), parameter **meos_39** = meosdata(ident = "HE", name = "helium", default_ref_state = "NBP", bibref = "Unpublished. Ortiz-Vega, Hall, Holste, Arp, Harvey and Lemmon (2018)", mw = 4.002602, tc = 5.1953, pc = 228.32, rhoc = 17.3837, ttr = 2.1768, ptr = 5.0393, t_nbp = 4.2238, tr = 5.1953, rhor = 17.3837, Rgas = 8.3144598, acf = -0.3836, t_max = 2000.0, p_max = 1000000.0, n_poly_eos = 6, n_exp_eos = 6, n_gauss_eos = 11, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.015559018,3.0638932,-4.2420844, 0.054418088,-0.18971904,0.087856262, 2.2833566,-0.53331595,-0.53296502, 0.99444915,-0.30078896,-1.6432563, 0.8029102,0.026838669,0.04687678, -0.14832766,0.03016211,-0.019986041, 0.14283514,0.007418269,-0.22989793, 0.79224829,-0.049386338,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.425,0.63, 0.69,1.83,0.575, 0.925,1.585,1.69, 1.51,2.9,0.8, 1.26,3.51,2.785, 1.0d0,4.22,0.83, 1.575,3.447,0.73, 1.634,6.13,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,2,2,3, 1,1,3,2,2,1, 2,1,2,1,1,3, 2,2,3,2,2,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,2,2,1,2,1, 2,2,2,2,2,2, 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -1.5497,-9.245,-4.76323,-6.3826,-8.7023,-0.255, -0.3523,-0.1492,-0.05,-0.1668,-42.2358,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -0.2471,-0.0983,-0.1556,-2.6782,-2.7077,-0.6621, -0.1775,-0.4821,-0.3069,-0.1758,-1357.6577,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 3.15,2.54505,1.2513,1.9416,0.5984,2.2282, 1.606,3.815,1.61958,0.6407,1.076,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.596,0.3423,0.761,0.9747,0.5868,0.5627, 2.5346,3.6763,4.5245,5.039,0.959,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,2, 2,2,2,2,2,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,2,2, 2,2,2,2,2,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 1, c_id = (/ 2.5,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 0.1733487932835764, a2_id = 0.4674522201550815)
- type(meosdata), parameter **meos_40** = meosdata(ident = "H2O", name = "water", default_ref_state = "TRIPLE_POINT", bibref = "DOI: 10.1063/1.1461829", mw = 18.015268, tc = 647.096, pc = 22064.0, rhoc = 17.8737279956, ttr = 273.16, ptr = 0.61248, t_nbp = 373.1243, tr = 647.096, rhor = 17.8737279956, Rgas = 8.314371357587, acf = 0.3443, t_max = 2000.0, p_max = 1000000.0, n_poly_eos = 7, n_exp_eos = 44, n_gauss_eos = 3, n_nona_eos = 2, n_assoc_eos = 0, n_eos = (/ 0.012533547935523,7.8957634722828,-8.7803203303561, 0.31802509345418,-0.26145533859358,-0.0078199751687981, 0.0088089493102134,-0.66856572307965,0.20433810950965, -0.66212605039687e-4,-0.19232721156002,-0.25709043003438, 0.16074868486251,-0.040092828925807,0.39343422603254e-6, -0.75941377088144e-5,0.00056250979351888,-0.15608652257135e-4, 0.11537996422951e-8,0.36582165144204e-6,-0.13251180074668e-11, -0.62639586912454e-9,-0.10793600908932,0.017611491008752, 0.22132295167546,-0.40247669763528,0.58083399985759, 0.0049969146990806,-0.031358700712549,-0.74315929710341, 0.47807329915480,0.020527940895948,-0.13636435110343, 0.014180634400617,0.008332650488071 3,-0.029052336009585, 0.038615085574206,-0.020393486513704,-0.0016554050063734, 0.0019955571979541,0.00015870308324157,-0.1638856834253e-4, 0.043613615723811,0.034994005463765,-0.076788197844621, 0.022446277332006,-0.62689710414685e-4,-0.55711118565645e-9, -0.19905718354408,0.31777497330738,-0.11841182425981, -31.306260323435,31.546140237781,-2521.3154341695 /), t_eos = (/ -0.5,0.875,1.0, 0.5,0.75,0.375, 1.0,4.0,6.0, 12.0,1.0,5.0, 4.0,2.0,13.0, 9.0,3.0,4.0, 11.0,4.0,13.0, 1.0,7.0,1.0, 9.0,10.0,10.0, 3.0,7.0,10.0, 10.0,6.0,10.0, 10.0,1.0,2.0, 3.0,4.0,8.0, 6.0,9.0,8.0, 16.0,22.0,23.0, 23.0,10.0,50.0, 44.0,46.0,50.0, 0.0,1.0,4.0 /), d_eos = (/ 1,1,1,2,2,3, 4,1,1,1,2,2, 3,4,4,5,7,9, 10,11,13,15,1,2, 2,2,3,4,4,4, 5,6,6,7,9,9, 9,9,9,10,10,12, 3,4,4,5,14,3, 6,6,6,3,3,3 /), l_eos = (/ 0,0,0,0,0,0, 0,1,1,1,1,1, 1,1,1,1,1,1, 1,1,1,1,2,2, 2,2,2,2,2,2, 2,2,2,2,2,2, 2,2,2,2,2,2, 3,3,3,3,4,6, 6,6,6,2,2,2 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -20.0,-20.0,-20.0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -150.0,-150.0,-250.0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.21,1.21,1.25,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 1.0,1.0,1.0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.

0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ -0.14874640856724,0.31806110878444,0.0d0 /), a_na = (/ 3.5,3.5,0.0d0 /), b_na = (/ 0.85,0.95,0.0d0 /), beta_na = (/ 0.3,0.3,0.0d0 /), big_a_na = (/ 0.32,0.32,0.0d0 /), big_b_na = (/ 0.2,0.2,0.0d0 /), big_c_na = (/ 28.0,32.0,0.0d0 /), big_d_na = (/ 700.0,800.0,0.0d0 /), n1_id = 1, n_id = 6, c_id = (/ 4.00632,0.012436,0.97315, 1.2795,0.96956,0.24873, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0 /), t_id = (/ 0.0,833.0,2289.0, 5009.0,5982.0,17800.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -8.3204464837678032, a2_id = 6.6832105275977254)

- type(meosdata), parameter **meos_41** = meosdata(ident = "R115", name = "r115", default_ref_state = "IIR", bibref = "DOI: 10.1021/acs.jced.5b00684", mw = 154.466416, tc = 353.1, pc = 3129.0, rhoc = 3.98, ttr = 173.75, ptr = 2.213, t_nbp = 233.932, tr = 353.1, rhor = 3.98, Rgas = 8.3144598, acf = 0.248, t_max = 550.0, p_max = 60000.0, n_poly_eos = 5, n_exp_eos = 7, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_↩
eos = 0, n_eos = (/ 1.20873,-3.54460,0.745302, 0.114128,4.36572e-4,0.988385, 1.13878,-0.0215633,-0.↩
630230, 0.0167901,-0.149412,-0.0271153, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0
/), t_eos = (/ 0.25,1.25,1.5, 0.25,0.875,2.375, 2.0,2.125,3.5, 6.5,4.75,12.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,3,7,1, 2,5,1,1,4,2, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,
0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,1, 1,1,2,2,2,3, 0,0,0,0,0,0, 0,0,0,0,0,0,
0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.↩
0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0
/), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /),
gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /),
epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /),
tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos
= (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0
/), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /),
big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 3, c_id = (/ 4.↩
0,7.142,10.61, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,289.0,1301.0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -13.4050037923356555, a2_id =
10.0015536023086682)
- type(meosdata), parameter **meos_42** = meosdata(ident = "R11", name = "r11", default_ref_state = "IIR", bibref = "DOI: 10.1016/0378-3812(92)87054-Q", mw = 137.368, tc = 471.11, pc = 4407.↩
638, rhoc = 4.032962, ttr = 162.68, ptr = 0.006510, t_nbp = 296.858, tr = 471.11, rhor = 4.↩
032962, Rgas = 8.314510, acf = 0.18875, t_max = 625.0, p_max = 30000.0, n_poly_eos =
14, n_exp_eos = 14, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 1.↩
25993633881,-2.60818574641,0.00982122542463, -1.06085385839,1.2282036351,0.118000776439, -
0.000698956926463,-0.0355428373358,0.00197169579643, -0.00848363012252,0.00417997567653,-
0.000242772533848, 0.00313371368974,0.396182646586e-5,0.339736319502, -0.203010634531,-0.↩
1060178599,0.45156488259, -0.339265767612,0.114338523359,0.0319537833995, 0.036790825978,-0.↩
961768948364e-5,0.00246717966418, -0.00167030256045,0.00240710110806,0.00156214678738, -0.↩
00323352596704,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.5,1.↩
5,5.0, 1.0,1.5,0.0, 5.0,2.0,3.0, 1.0,2.0,4.0, 1.0,4.0,5.0, 6.0,3.5,5.5, 7.5,3.0,2.5, 5.0,1.5,11.0, 9.0,13.0,5.0,
9.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,2,3, 3,4,4,5,5,5,
6,8,1,1,2,2, 2,3,4,6,10,3, 5,8,9,9,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0,
0,0,0,0,0,0, 0,0,2,2,2,2, 2,2,2,2,2,4, 6,6,6,6,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_↩
eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0,
1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,

0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_↩ exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 2, n_id = 8, c_id = (/ 4.00564923,0.0002228875,1.0, 2.0,1.0,2.0, 1.0,2.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,1.0,1561.076, 1218.647,770.035,572.634, 502.854,346.746,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -17.7938110160538514, a2_id = 10.0839710927509554)

- type(meosdata), parameter **meos_43** = meosdata(ident = "13BD", name = "13butadiene", default_ref_state = "NBP", bibref = "Unpublished. Gao, Wu and Lemmon (2017)", mw = 54.09044, tc = 425.135, pc = 4305.3, rhoc = 4.53, ttr = 164.25, ptr = 0.069922, t_nbp = 268.661, tr = 425.135, rhor = 4.53, Rgas = 8.3144598, acf = 0.192, t_max = 426.0, p_max = 10000.0, n_poly_eos = 5, n_exp_eos = 7, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.84958959,-2.028566,0.03757012, 0.06081274,0.000204315,0.066043135, 0.42974178,0.004085874,-0.25316, 0.02068683,-0.041815041,-0.02051974, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.25,1.5, 0.25,0.875,2.375, 2.0,2.125,3.5, 6.5,4.75,12.5, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,3,7,1, 2,5,1,1,4,2, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,1, 1,1,2,2,2,3, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.↩ 0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.↩ 0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 4, c_id = (/ 4.0,2.3797,11.213, 8.1824,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0 /), t_id = (/ 0.0,308.0,1210.0, 2631.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -1.0260215656338829, a2_id = 2.7310597807829633)
- type(meosdata), parameter **meos_44** = meosdata(ident = "ACETONE", name = "acetone", default_ref_↩ state = "NBP", bibref = "DOI: 10.1021/je050186n", mw = 58.07914, tc = 508.1, pc = 4692.4, rhoc = 4.7, ttr = 178.5, ptr = 0.002326, t_nbp = 329.22, tr = 508.1, rhor = 4.7, Rgas = 8.314472, acf = 0.3071, t_max = 550.0, p_max = 700000.0, n_poly_eos = 5, n_exp_eos = 7, n_gauss_eos = 0, n_nona_eos = 0, n_assoc↩ _eos = 0, n_eos = (/ 0.90041,-2.1267,-0.083409, 0.065683,0.00016527,-0.039663, 0.72085,0.0092318,-0.↩ 17217, -0.14961,-0.076124,-0.018166, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.25,1.5, 0.25,0.875,2.375, 2.0,2.125,3.5, 6.5,4.75,12.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,3,7,1, 2,5,1,1,4,2, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,1, 1,1,2,2,2,3, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.↩ 0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.↩

0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 4, c_id = (/ 4.0,3.↩ 7072,7.0675, 11.012,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,310.0,3480.0, 1576.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 0.3015070258580721, a2_id = 2.7317365462703083)

- type(meosdata), parameter **meos_45** = meosdata(ident = "NH3", name = "ammonia", default_ref_state = "TRIPLE_POINT", bibref = "Gao et al. (2023). doi:10.1063/5.0128269", mw = 17.03052, tc = 405.56, pc = 11363.4, rhoc = 13.696, ttr = 195.49, ptr = 6.05438, t_nbp = 239.832, tr = 405.56, rhor = 13.↩ 696, Rgas = 8.3144598, acf = 0.256, t_max = 725.0, p_max = 1000000.0, n_poly_eos = 5, n_exp_↩ eos = 3, n_gauss_eos = 12, n_nona_eos = 0, n_assoc_eos = 2, n_eos = (/ 0.006132232,1.7395866,-2.2261792, -0.30127553,0.08967023,-0.076387037, -0.84063963,-0.27026327,6.212578, -5.7844357,2.↩ 4817542,-2.3739168, 0.01493697,-3.7749264,0.0006254348, -0.000017359,-0.13462033,0.07749072839, -1.6909858,0.93739074,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.382,1.0, 1.0,0.677,2.915, 3.51,1.063,0.655, 1.3,3.1,1.4395, 1.623,0.643,1.13, 4.5,1.0,4.0, 4.3315,4.015,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,2,3,3, 2,3,1,1,1,2, 2,1,3,3,1,1, 1,1,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,2, 2,1,2,2,2,2, 2,2,2,2,2,2, 2,2,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.↩ 0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -0.42776,-0.6424,-0.8175,-0.7995,-0.91,-0.3574, -1.21,-4.14,-22.↩ 56,-22.68,-2.8452,-2.8342, 0.0d0,0.0d0 /), beta_eos = (/ -1.708,-1.4865,-2.0915,-2.43,-0.488,-1.1, -0.85,-1.↩ 14,-945.64,-993.85,0.3696,0.2962, 0.0d0,0.0d0 /), gamma_eos = (/ 1.036,1.2777,1.083,1.2906,0.928,0.934, 0.919,1.852,1.05897,1.05277,1.108,1.313, 0.0d0,0.0d0 /), epsilon_eos = (/ -0.0726,-0.1274,0.7527,0.57,2.↩ 2,-0.243, 2.96,3.02,0.9574,0.9576,0.4478,0.44689, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,2, 2,2,2,2,2,2, 0,0 /), del_exp_eos = (/ 2,2,2,2,2,2, 2,2,2,2,2,2, 0,0 /), b_assoc_eos = (/ 1.244,0.6826 /), n_na = (/ 0.0d0,0.↩ 0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 4, c_id = (/ 4.0,2.224,3.148, 0.9579,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0d0,1646.0d0,3965.0d0, 7231.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -6.59406093943886, a2_id = 5.60101151987913)
- type(meosdata), parameter **meos_46** = meosdata(ident = "BENZENE", name = "benzene", default_ref_↩ state = "NBP", bibref = "Unpublished. Thol, Lemmon, and Span (2012).", mw = 78.11184, tc = 562.02, pc = 4907.277, rhoc = 3.901, ttr = 278.674, ptr = 4.785, t_nbp = 353.219, tr = 562.02, rhor = 3.901, Rgas = 8.314472, acf = 0.211, t_max = 725.0, p_max = 500000.0, n_poly_eos = 5, n_exp_eos = 5, n_gauss_eos = 4, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.03512459,2.2338,-3.10542612, -0.577233,0.25101,-0.705518, -0.139648,0.83494,-0.331456, -0.0279953,0.7099766,-0.3732185, -0.0629985,-0.803041,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.29,0.696, 1.212,0.595,2.51, 3.96,1.24,1.83, 0.↩ 82,0.57,2.04, 3.2,0.78,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,2,3,1, 3,2,2,7,1,1, 3,3,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,2, 2,1,2,1,2,2, 2,2,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -1.032,-1.423,-1.071,-14.2,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -1.864,-1.766,-1.825,-297.9,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.118,0.639,0.654,1.164,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.729,0.907,0.765,0.870,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), del_↩ exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /),

a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 4, c_id = (/ 3.94645,7.36374,18.649, 4.01834,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,4116.0,1511.0, 630.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -0.673584364528315, a2_id = 2.5555552197611604)

- type(meosdata), parameter **meos_47** = meosdata(ident = "N2O", name = "n2o", default_ref_state = "NBP", bibref = "DOI: 10.1021/je050186n", mw = 44.0128, tc = 309.52, pc = 7245.0, rhoc = 10.27, ttr = 182.↩
33, ptr = 87.84, t_nbp = 184.68, tr = 309.52, rhor = 10.27, Rgas = 8.314472, acf = 0.162, t_max = 525.0, p_max = 50000.0, n_poly_eos = 5, n_exp_eos = 7, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.88045,-2.4235,0.38237, 0.068917,0.00020367,0.13122, 0.46032,-0.0036985,-0.23263, -0.↩
00042859,-0.042810,-0.023038, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.25,1.5, 0.25,0.875,2.375, 2.0,2.125,3.5, 6.5,4.75,12.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,3,7,1, 2,5,1,1,4,2, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,1, 1,1,2,2,2,3, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.↩
0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.↩
0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 4, c_id = (/ 3.5,2.1769,1.6145, 0.48393,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,879.0,2372.0, 5447.↩
0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -4.4262724194662564, a2_id = 4.↩
3120468016770888)
- type(meosdata), parameter **meos_48** = meosdata(ident = "CO2", name = "co2", default_ref_↩
state = "IIR", bibref = "DOI: 10.1063/1.555991", mw = 44.0098, tc = 304.1282, pc = 7377.3, rhoc = 10.6249063, ttr = 216.592, ptr = 517.95, t_nbp = 194.686, tr = 304.1282, rhor = 10.6249063, Rgas = 8.31451, acf = 0.22394, t_max = 2000.0, p_max = 800000.0, n_poly_eos = 7, n_exp_↩
eos = 27, n_gauss_eos = 5, n_nona_eos = 3, n_assoc_eos = 0, n_eos = (/ 0.388568232032,2.↩
93854759427,-5.58671885349, -0.767531995925,0.317290055804,0.548033158978, 0.122794112203,2.↩
16589615432,1.58417351097, -0.231327054055,0.0581169164314,-0.553691372054, 0.489466159094,-0.0242757398435,0.0624947905017, -0.121758602252,-0.370556852701,-0.0167758797004, -0.↩
11960736638,-0.0456193625088,0.0356127892703, -0.00744277271321,-0.00173957049024,-0.↩
0218101212895, 0.0243321665592,-0.0374401334235,0.143387157569, -0.134919690833,-0.0231512250535,0.↩
0123631254929, 0.00210583219729,-0.000339585190264,0.00559936517716, -0.000303351180556,-213.654886883,26641.5691493, -24027.2122046,-283.41603424,212.472844002, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.0,0.75,1.0, 2.↩
0,0.75,2.0, 0.75,1.5,1.5, 2.5,0.0,1.5, 2.0,0.0,1.0, 2.0,3.0,6.0, 3.0,6.0,8.0, 6.0,0.0,7.0, 12.0,16.0,22.0, 24.↩
0,16.0,24.0, 8.0,2.0,28.0, 14.0,1.0,0.0, 1.0,3.0,3.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,1,2,2, 3,1,2,4,5,5, 5,6,6,6,1,1, 4,4,4,7,8,2, 3,3,5,5,6,7, 8,10,4,8,2,2, 2,3,3,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,1,1,1,1,1, 1,1,1,1,2,2, 2,2,2,2,2,3, 3,3,4,4,4,4, 4,4,5,6,2,2, 2,2,2,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.↩
0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), eta_eos = (/ -25.0,-25.0,-25.0,-15.0,-20.0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0 /), beta_eos = (/ -325.0,-300.0,-300.0,-275.0,-275.0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.16,1.19,1.19,1.25,1.22,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0 /), epsilon_eos = (/ 1.0,1.0,1.0,1.0,1.0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0

/), tau_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), b↩
_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ -0.666422765408,0.726086323499,0.0550686686128 /), a_na = (/ 3.5,3.5,3.0 /), b_na = (/ 0.875,0.925,0.875 /), beta_na = (/ 0.3,0.3,0.3 /), big_a_na = (/ 0.7,0.7,0.7 /), big_b_na = (/ 0.3,0.3,1.0 /), big_c_na = (/ 10.0,10.0,12.5 /), big_d_na = (/ 275.0,275.0,275.0 /), n1_id = 1, n_id = 6, c_id = (/ 3.5,1.99427042,0.621052475, 0.411952928,1.04028922,0.0832767753, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,958.49956,1858.80115, 2061.10114,3443.89908,8238.20035, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -6.124871063353245, a2_id = 5.1155963185961815)

- type(meosdata), parameter **meos_49** = meosdata(ident = "R14", name = "r14", default_ref_state = "TRIPLE_POINT", bibref = "ISBN: 978-3-662-02610-6", mw = 88.01, tc = 227.51, pc = 3750.0, rhoc = 7.1094194, ttr = 89.54, ptr = 0.1012, t_nbp = 145.10, tr = 227.51, rhor = 7.1094194, Rgas = 8.31451, acf = 0.1785, t_max = 623.0, p_max = 51000.0, n_poly_eos = 16, n_exp_eos = 6, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ -0.334698748966,0.586690904687,-0.147068929692, 1.03999039623,-2.45792025288,0.799614557889, -0.749498954929,0.152177772502,-0.293408331764, 0.717794502866,-0.0426467444199,0.226562749365, -0.391091694003,-0.0257394804936,0.0554844884782, 0.00610988261204,0.↩
334698748966,-0.586690904687, 0.147068929692,-0.190315426142,0.716157133959, -0.703161904626,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_↩
eos = (/ 3.0d0,4.0d0,5.0d0, 0.0d0,1.0d0,2.0d0, 3.0d0,4.0d0,0.0d0, 1.0d0,2.0d0,0.0d0, 1.0d0,0.0d0,1.0d0, 1.0d0,3.0d0,4.0d0, 5.0d0,3.0d0,4.0d0, 5.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 0,0,0,1,1,1, 1,1,2,2,2,3, 3,4,4,5,0,0, 0,2,2,2,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,2,2, 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.↩
0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.99832625,0.99832625, 0.99832625,0.99832625,0.99832625, 0.99832625,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.↩
0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 5, n_id = 5, c_id = (/ 3.9465247,-0.0088586725,0.00013939626, -0.30056204e-6,0.20504001e-9,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,1.0,2.↩
0, 3.0,4.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -8.5391328033953542, a2_id = 13.4878238763322749)
- type(meosdata), parameter **meos_50** = meosdata(ident = "MEG", name = "eglycol", default_ref_state = "NBP", bibref = "Unpublished. Zhou and Lemmon (2018)", mw = 62.06784, tc = 719.0, pc = 10508.7, rhoc = 5.88, ttr = 260.6, ptr = 0.0002366, t_nbp = 470.313, tr = 719.0, rhor = 5.88, Rgas = 8.3144598, acf = 0.619, t_max = 750.0, p_max = 100000.0, n_poly_eos = 7, n_exp_eos = 7, n_gauss_eos = 7, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.019393376,1.2215576,1.2751617, -3.6681302,-1.4660821,0.24628603, -0.063217756,1.4131488,3.5245547, -2.2658015,0.94972119,-0.13037392, 0.19881857,-0.022141839,1.↩
1273408, -0.12188623,-0.79487875,-0.024231918, -0.00574040155,0.0083087704,-0.041852456, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.1,1.27, 1.244,1.1,0.32, 1.0,0.89,1.2, 1.34,1.3,1.49, 1.23,0.18,1.1, 0.75,0.79,0.77, 0.↩
6,1.0,1.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), d_eos = (/ 5,1,1,1,2,3, 4,2,3,3,4,5, 6,7,1,1,2,3, 4,5,3,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,1,1,1,1,1, 1,1,2,2,2,2, 2,2,2,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -0.9,-1.35,-0.8,-1.9,-2.0,-1.3, -20.0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -0.91,-1.25,-0.97,-0.5,-1.↩

0,-0.42, -1000.0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.17,1.49,1.3,1.5,1.2,1.2, 1.07,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 1.37,0.4,1.0d0,2.45,1.9,2.0d0, 0.9,0.↩ 0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,2, 2,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,2,2, 2,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 2, c_id = (/ 4.0,20.86,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,1260.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -1.2230958209345886, a2_id = 3.8389800933704281)

- type(meosdata), parameter **meos_51** = meosdata(ident = "NC7", name = "heptane", default_ref_state = "NBP", bibref = "Unpublished. Tenji, Thol, Lemmon and Span (2018).Fundamental Equation of State f", mw = 100.202, tc = 540.2, pc = 2735.73, rhoc = 2.33, ttr = 182.55, ptr = 0.00017426, t_nbp = 371.550, tr = 540.2, rhor = 2.33, Rgas = 8.3144598, acf = 0.349, t_max = 600.0, p_max = 100000.0, n_poly_eos = 5, n_exp_eos = 5, n_gauss_eos = 4, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.04021974,1.417638,-1.822198, -0.8536007,0.265174,-1.968992, -1.420553,0.8501985,-1.516957, -0.02581013,2.810547,-0.008215312, -0.8616429,-0.2926342,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.208,0.87, 1.036,0.↩ 574,2.0, 2.26,1.1,1.87, 0.985,0.73,1.13, 1.23,1.4,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,2,3,1, 3,2,2,7,1,1, 3,2,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,2, 2,1,2,1,2,2, 2,2,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -1.017,-5.3,-1.135,-1.227,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -1.41,-83.0,-1.54,-1.405,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.35,1.14,1.04,1.26,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.9,0.92,0.52,1.236,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.↩ 0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.↩ 0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 4, c_id = (/ 4.0,11.36,18.82, 29.14,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,190.0,3800.0, 1500.0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 17.46311465657638, a2_id = -2.9932383994019123)
- type(meosdata), parameter **meos_52** = meosdata(ident = "NC4", name = "butane", default_ref↩ _state = "IIR", bibref = "DOI: 10.1063/1.1901687", mw = 58.1222, tc = 425.125, pc = 3796.↩ 0, rhoc = 3.922769613, ttr = 134.895, ptr = 0.0006656, t_nbp = 272.660, tr = 425.125, rhor = 3.922769613, Rgas = 8.314472, acf = 0.201, t_max = 575.0, p_max = 200000.0, n_poly_↩ eos = 7, n_exp_eos = 16, n_gauss_eos = 2, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 2.↩ 5536998241635,-4.4585951806696,0.82425886369063, 0.11215007011442,-0.035910933680333,0.↩ 016790508518103, 0.032734072508724,0.95571232982005,-1.0003385753419, 0.085581548803855,-0.↩ 025147918369616,-0.0015202958578918, 0.004706068232642,-0.097845414174006,-0.04831790415876, 0.17841271865468,0.018173836739334,-0.11399068074953, 0.019329896666669,0.001157587740101,0.↩ 0001525380869811 6, -0.043688558458471,-0.0082403190629989,-0.028390056949441, 0.0014904666224681,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 5,1.0,1.↩ 5, 0.0,0.5,0.5, 0.75,2.0,2.5, 2.5,1.5,1.0, 1.5,4.0,7.0, 3.0,7.0,3.0, 1.0,6.0,0.0, 6.0,13.0,2.0, 0.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,4, 4,1,1,2,7,8, 8,1,2,3,3,4, 5,5,10,2,6,1, 2,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,1,1,1,1,1, 1,2,2,2,2,2, 2,2,2,3,3,2, 2,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_↩ eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -10.0,-10.0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.↩

0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -150.0,-200.0,0.0d0,0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.16,1.13,0.0d0,0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.85,1.0,0.0d0,0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,0,0,0,0, 0,0,0,0,0,0, 0,0 /),
del_exp_eos = (/ 2,2,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.↩
0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0
/), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.↩
0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/ 4.24680487,5.54913289,11.↩
4648996, 7.59987584,9.66033239,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,329.↩
40404,1420.17366, 2113.08938,4240.8573,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id =
-5.4249697888268713, a2_id = 4.9194999371032626)

- type(meosdata), parameter **meos_53** = meosdata(ident = "OXYL", name = "oxylene", default_ref_state =
"NBP", bibref = "DOI: 10.1063/1.3703506", mw = 106.165, tc = 630.259, pc = 3737.5, rhoc = 2.6845, ttr
= 247.985, ptr = 0.0228, t_nbp = 417.521, tr = 630.259, rhor = 2.6845, Rgas = 8.314472, acf = 0.312,
t_max = 700.0, p_max = 70000.0, n_poly_eos = 7, n_exp_eos = 5, n_gauss_eos = 4, n_nona_eos = 0, n↩
_assoc_eos = 0, n_eos = (/ 0.0036765156,-0.13918171,0.014104203, 1.5398899,-2.3600925,-0.44359159,
0.19596977,-1.0909408,-0.21890801, 1.1179223,-0.93563815,-0.018102996, 1.4172368,-0.57134695,-0.↩
081944041, -40.682878,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.6,0.91, 0.3,0.895,1.↩
167, 0.435,2.766,3.8, 1.31,3.0,0.77, 1.41,4.8,1.856, 2.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/
5,1,4,1,1,2, 3,1,3,2,2,7, 1,1,3,3,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0
/), l_eos = (/ 0,0,0,0,0,0, 0,2,2,1,2,1, 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,
0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -1.1723,-1.095,-1.6166,-20.4,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -2.442,-1.342,-3.0,-450.0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.2655,0.3959,0.7789,1.↩
162,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.552,0.728,0.↩
498,0.894,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,0,0,
0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_↩
na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/
0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/
0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/ 3.748798,4.754892,6.↩
915052, 25.84813,10.93886,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,225.0,627.0,
1726.0,4941.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 10.1373795661858708, a2_id
= -0.9128323735238781)
- type(meosdata), parameter **meos_54** = meosdata(ident = "R134A", name = "r134a", default_ref_state = "IIR",
bibref = "DOI: 10.1063/1.555958", mw = 102.032, tc = 374.21, pc = 4059.28, rhoc = 5.017053, ttr = 169.85,
ptr = 0.3896, t_nbp = 247.076, tr = 374.18, rhor = 4.978830171, Rgas = 8.314471, acf = 0.32684, t_max =
455.0, p_max = 70000.0, n_poly_eos = 8, n_exp_eos = 13, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_↩
eos = 0, n_eos = (/ 0.05586817,0.498223,0.02458698, 0.0008570145,0.0004788584,-1.800808, 0.2671641,-
0.04781652,0.01423987, 0.3324062,-0.007485907,0.0001017263, -0.5184567,-0.08692288,0.2057144, -
0.005000457,0.0004603262,-0.003497836, 0.006995038,-0.01452184,-0.0001285458, 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ -0.↩
5,0.0,0.0, 0.0,1.5,1.5, 2.0,2.0,1.0, 3.0,5.0,1.0, 5.0,5.0,6.0, 10.0,10.0,10.0, 18.0,22.0,50.0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/
2,1,3,6,6,1, 1,2,5,2,2,4, 1,4,1,2,4,1, 5,3,10,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0
/), l_eos = (/ 0,0,0,0,0,0, 0,0,1,1,1,2, 2,2,2,2,2,3, 3,3,4,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,
0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.↩
0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.↩

0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.↩ 0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_↩ na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 3, n_id = 3, c_id = (/ -0.629789,0.↩ 3770180792987904,0.06058548894986743, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0 /), t_id = (/ 0.0,0.5,0.75, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -1.0195506821162033, a2_id = 9.0471440883402678)

- type(meosdata), parameter **meos_55** = meosdata(ident = "C3", name = "propane", default_ref_state = "IIR", bibref = "DOI: 10.1021/je900217v", mw = 44.09562, tc = 369.89, pc = 4251.2, rhoc = 5.0, ttr = 85.525, ptr = 0.000000172, t_nbp = 231.036, tr = 369.89, rhor = 5.0, Rgas = 8.314472, acf = 0.1521, t_max = 650.0, p_max = 1000000.0, n_poly_eos = 5, n_exp_eos = 6, n_gauss_eos = 7, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.042910051,1.7313671,-2.4516524, 0.34157466,-0.46047898,-0.66847295, 0.20889705,0.19421381,-0.22917851, -0.60405866,0.066680654,0.017534618, 0.33874242,0.22228777,-0.23219062, -0.09220694,-0.47575718,-0.017486824, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.33,0.8, 0.43,0.90,2.46, 2.09,0.88,1.↩ 09, 3.25,4.62,0.76, 2.50,2.75,3.05, 2.55,8.40,6.75, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,2,2,1, 3,6,6,2,3,1, 1,1,2,2,4,1, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,1, 1,1,1,2,2,2, 2,2,2,2,2,2, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -0.963,-1.977,-1.917,-2.307,-2.546,-3.28, -14.↩ 6,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -2.33,-3.47,-3.15,-3.19,-0.92,-18.8, -547.↩ 8,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.684,0.829,1.419,0.817,1.5,1.426, 1.↩ 093,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 1.283,0.6936,0.788,0.473,0.8577,0.271, 0.948,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,2, 2,0,0,0,0,0, 0,0 /), del_↩ exp_eos = (/ 2,2,2,2,2,2, 2,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/ 4.0,3.043,5.874, 9.337,7.922,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,393.0,1237.0, 1984.0,4351.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0 /), a1_id = -4.9705912033525124, a2_id = 4.293524424315649)
- type(meosdata), parameter **meos_56** = meosdata(ident = "TOLU", name = "toluene", default_ref_state = "NBP", bibref = "DOI: 10.1021/je050186n", mw = 92.13842, tc = 591.75, pc = 4126.3, rhoc = 3.169, ttr = 178.0, ptr = 0.00003939, t_nbp = 383.75, tr = 591.75, rhor = 3.169, Rgas = 8.314472, acf = 0.2657, t_max = 700.0, p_max = 500000.0, n_poly_eos = 6, n_exp_eos = 6, n_gauss_eos = 0, n_nona_eos = 0, n_assoc↩ _eos = 0, n_eos = (/ 0.96464,-2.7855,0.86712, -0.18860,0.11804,0.00025181, 0.57196,-0.029287,-0.43351, -0.12540,-0.028207,0.014076, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.125,1.5, 1.375,0.25,0.875, 0.625,1.75,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.↩ 0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos =

(/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.↩ 0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 6, c_id = (/ 4.0,1.6994,8.0577, 17.059,8.4567,8.6423, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,190.0,797.0, 1619.↩ 0,3072.0,7915.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 3.5241204689061192, a2_id = 1.1360804066266901)

- type(meosdata), parameter **meos_57** = meosdata(ident = "R142B", name = "r142b", default_ref_state = "IIR", bibref = "DOI: 10.1021/je050186n", mw = 100.49503, tc = 410.26, pc = 4055.0, rhoc = 4.438, ttr = 142.↩ 72, ptr = 0.003632, t_nbp = 264.03, tr = 410.26, rhor = 4.438, Rgas = 8.314472, acf = 0.2321, t_max = 470.0, p_max = 60000.0, n_poly_eos = 5, n_exp_eos = 7, n_gauss_eos = 0, n_nona_eos = 0, n_assoc↩ _eos = 0, n_eos = (/ 1.0038,-2.7662,0.42921, 0.081363,0.00024174,0.48246, 0.75542,-0.007430,-0.41460, -0.016558,-0.10644,-0.021704, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.25,1.5, 0.25,0.875,2.375, 2.0,2.125,3.5, 6.5,4.75,12.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,3,7,1, 2,5,1,1,4,2, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,1, 1,1,2,2,2,3, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.↩ 0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.↩ 0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/ 4.0,5.0385,6.8356, 4.0591,2.8136,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,473.0,1256.0, 2497.↩ 0,6840.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -12.6016704500950851, a2_id = 8.316026972375175)
- type(meosdata), parameter **meos_58** = meosdata(ident = "NC5", name = "pentane", default_ref_state = "NBP", bibref = "Unpublished. Thol, Uhde, Lemmon and Span (2019). Fundamental Equations of State ", mw = 72.14878, tc = 469.7, pc = 3367.5, rhoc = 3.21, ttr = 143.47, ptr = 0.000078028, t_nbp = 309.209, tr = 469.7, rhor = 3.21, Rgas = 8.3144598, acf = 0.251, t_max = 650.0, p_max = 780000.0, n_poly_eos = 5, n_exp_eos = 6, n_gauss_eos = 5, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.042952795,2.4923999,-2.603872, -0.83829913,0.19223378,-3.0778196, -0.000324816,-1.6781976,0.6416425, -1.7300934,-0.017585046,4.↩ 5708883, -0.0758188,-0.62122633,-0.42413043, -2.0418443,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t↩ _eos = (/ 1.0d0,0.367,0.704, 1.04,0.494,1.34, 0.688,1.688,0.88, 1.357,1.021,0.979, 2.966,1.35,0.664, 0.↩ 937,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,2,3,1, 1,3,2,2,7,1, 1,3,2,2,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,2, 3,2,1,2,1,2, 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.↩ 0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0 /), eta_eos = (/ -1.01,-4.77,-1.13,-1.08,-1.12,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -0.583,-31.6,-0.52,-0.654,-0.75,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.06,1.37,1.09,1.19,0.83,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.↩

0d0,0.0d0 /), epsilon_eos = (/ 0.927,0.968,0.735,1.196,0.617,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,2,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_↩na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 4, c_id = (/ 4.0,6.618,15.97, 15.29,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,154.0,1324.0, 2634.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 8.5092528322783494, a2_id = 0.0643058406269243)

- type(meosdata), parameter **meos_59** = meosdata(ident = "3MP", name = "3methylpentane", default_ref_↩state = "NBP", bibref = "Unpublished. Gao, Wu and Lemmon (2017)", mw = 86.17536, tc = 506.0, pc = 3184.5, rhoc = 2.78, ttr = 110.263, ptr = 0.0000000002, t_nbp = 336.379, tr = 506.0, rhor = 2.78, Rgas = 8.3144598, acf = 0.268, t_max = 550.0, p_max = 1000000.0, n_poly_eos = 5, n_exp_eos = 5, n_gauss_eos = 6, n_nona↩_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.006178288,0.763315017,-0.5546657, -1.0604327,0.23117181,-1.8757299, -0.9327912,1.0720552,-0.2830806, -0.024600061,0.87360786,0.008687374, -0.27160944,-0.↩12365512,-0.12052593, -0.53359397,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0d0,0.16,1.0d0, 1.0d0,0.386,1.54, 2.0d0,1.0d0,2.5, 1.66,0.44,1.0d0, 0.55,0.705,1.5, 1.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 5,1,1,2,3,1, 3,2,2,8,1,1, 3,2,1,3,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,2, 2,1,2,2,2,2, 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.↩0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -1.409,-2.53,-1.↩781,-2.045,-0.688,-20.1, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -1.876,-1.158,-1.808,-1.646,-1.0d0,-660.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.↩2603,1.207,1.045,1.069,0.923,1.109, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.7065,2.19,0.244,1.014,0.689,0.905, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,2,2, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,2,2, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.↩0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 4, c_id = (/ 7.0,17.↩741,25.090, 7.4047,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,3946.0,1555.0, 470.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 4.6479377291134423, a2_id = -0.9065150384577076)
- type(meosdata), parameter **meos_60** = meosdata(ident = "NC9", name = "nonane", default_ref_state = "NBP", bibref = "DOI: 10.1021/je050186n", mw = 128.2551, tc = 594.55, pc = 2281.0, rhoc = 1.81, ttr = 219.7, ptr = 0.0004444, t_nbp = 423.91, tr = 594.55, rhor = 1.81, Rgas = 8.314472, acf = 0.4433, t_max = 600.0, p_max = 800000.0, n_poly_eos = 6, n_exp_eos = 6, n_gauss_eos = 0, n_nona_eos = 0, n_assoc↩_eos = 0, n_eos = (/ 1.1151,-2.7020,0.83416, -0.38828,0.13760,0.00028185, 0.62037,0.015847,-0.61726, -0.15043,-0.012982,0.0044325, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.125,1.5, 1.375,0.25,0.875, 0.625,1.75,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.↩0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos =

(/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.↩
0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/ 17.349,24.926,24.842, 11.188,17.483,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,1221.0,2244.0, 5008.↩
0,11724.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 10.792727988975912, a2_id = -8.2418358074837137)

- type(meosdata), parameter **meos_61** = meosdata(ident = "XE", name = "xenon", default_ref_state = "NBP", bibref = "DOI: 10.1021/je050186n", mw = 131.293, tc = 289.733, pc = 5842.0, rhoc = 8.4, ttr = 161.405, ptr = 81.77, t_nbp = 165.05, tr = 289.733, rhor = 8.4, Rgas = 8.314472, acf = 0.00363, t_max = 750.0, p_max = 700000.0, n_poly_eos = 6, n_exp_eos = 6, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.83115,-2.3553,0.53904, 0.014382,0.066309,0.00019649, 0.14996,-0.035319,-0.15929, -0.027521,-0.↩
023305,0.0086941, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.25,1.125,1.↩
5, 1.375,0.25,0.875, 0.625,1.75,3.625, 3.625,14.5,12.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), d_eos = (/ 1,1,1,2,3,7, 2,5,1,4,3,4, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,2,2,3,3, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.↩
0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.↩
0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 1, c_id = (/ 2.↩
5,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -3.8227169849622977, a2_id = 3.8416390607135864)
- type(meosdata), parameter **meos_62** = meosdata(ident = "NC11", name = "c11", default_ref_state = "NBP", bibref = "DOI: 10.1134/S0040601511080027", mw = 156.30826, tc = 638.8, pc = 1990.4, rhoc = 1.5149, ttr = 247.541, ptr = 0.0004461, t_nbp = 468.934, tr = 638.8, rhor = 1.5149, Rgas = 8.314472, acf = 0.539, t_max = 700.0, p_max = 500000.0, n_poly_eos = 6, n_exp_eos = 8, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ -0.66172706,1.3375396,-2.5608399, 0.10678910,0.↩
00028873614,0.049587209, 0.55407101e-7,0.99754712,1.5774025, 0.0013108354,-0.59326961,-0.↩
093001876, -0.17960228,-0.022560853,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.5,0.25,1.25, 0.25,0.875,1.375, 0.0,2.375,2.0d0, 2.125,3.5,6.5, 4.75,12.5,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 1,1,1,3,7,2, 1,1,2,5,1,1, 4,2,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,1,1,1,2,2, 2,3,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.↩
0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.↩

0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 4, c_id = (/ 5.9624,20.↩ 584,44.512, 16.520,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,323.0,1597.0, 3302.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 32.12928483616682, a2_id = -10.75942326931605)

- type(meosdata), parameter **meos_63** = meosdata(ident = "EBZN", name = "ebenzene", default_ref_state = "NBP", bibref = "DOI: 10.1063/1.3703506", mw = 106.165, tc = 617.12, pc = 3622.4, rhoc = 2.741016, ttr = 178.2, ptr = 0.000004002, t_nbp = 409.314, tr = 617.12, rhor = 2.741016, Rgas = 8.314472, acf = 0.305, t_max = 700.0, p_max = 60000.0, n_poly_eos = 7, n_exp_eos = 5, n_gauss_eos = 4, n_nona_↩ eos = 0, n_assoc_eos = 0, n_eos = (/ 0.0018109418,-0.076824284,0.041823789, 1.5059649,-2.4122441,-0.47788846, 0.18814732,-1.0657412,-0.20797007, 1.1222031,-0.99300799,-0.027300984, 1.3757894,-0.↩ 44477155,-0.07769742, -2.16719,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,1.0,0.92, 0.27,0.↩ 962,1.033, 0.513,2.31,3.21, 1.26,2.29,1.0, 0.6,3.6,2.1, 0.5,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 5,1,4,1,1,2, 3,1,3,2,2,7, 1,1,3,3,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,2,2,1,2,1, 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -1.178,-1.07,-1.775,-15.45,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -2.437,-1.488,-4.0,-418.6,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.2667,0.4237,0.8573,1.↩ 15,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.5494,0.7235,0.↩ 493,0.8566,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.↩ 0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.↩ 0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 4, c_id = (/ 5.2557889,9.7329909,11.↩ 201832, 25.440749,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,585.0,4420.0, 1673.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 5.7040936889063971, a2_id = -0.5241459501533468)
- type(meosdata), parameter **meos_64** = meosdata(ident = "R143A", name = "r143a", default_ref_state = "IIR", bibref = "DOI: 10.1063/1.1318909", mw = 84.041, tc = 345.857, pc = 3761.0, rhoc = 5.↩ 12845, ttr = 161.34, ptr = 1.0749, t_nbp = 225.91, tr = 345.857, rhor = 5.12845, Rgas = 8.314472, acf = 0.2615, t_max = 650.0, p_max = 100000.0, n_poly_eos = 5, n_exp_eos = 12, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 7.7736443,-8.70185,-0.27779799, 0.14609220,0.↩ 0089581616,-0.20552116, 0.10653258,0.023270816,-0.013247542, -0.04279387,0.36221685,-0.25671899, -0.092326113,0.083774837,0.017128445, -0.01725611,0.0049080492,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.67,0.833,1.7, 1.82,0.35,3.9, 0.95,0.0,1.19, 7.2,5.9,7.65, 7.5,7.45,15.5, 22.0,19.0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0 /), d_eos = (/ 1,1,1,2,5,1, 3,5,7,1,2,2, 3,4,2,3,5,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,1, 1,1,1,2,2,2, 2,2,3,3,3,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.↩ 0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.↩ 0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.↩ 0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.↩

0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 3, c_id = (/ 1.0578,4.↩
4402,3.7515, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.33,1791.0,823.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 5.9030332880254637, a2_id = 7.3072591981034742)

- type(meosdata), parameter **meos_65** = meosdata(ident = "C3_1", name = "cyclopro", default_ref_↩
state = "IIR", bibref = "Polt, Platzer and Maurer (1992). Parameter der Thermischen Zustandsgle-ichung von", mw = 42.081, tc = 398.3, pc = 5579.7, rhoc = 6.1429149, ttr = 145.7, ptr = 0.07, t↩
_nbp = 241.670, tr = 398.3, rhor = 6.1429149, Rgas = 8.3143, acf = 0.1305, t_max = 473.0, p_↩
max = 28000.0, n_poly_eos = 16, n_exp_eos = 6, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ -1.37016097588,2.12444673002,-0.578908942724, -1.15633726379,2.52574014413,-2.↩
82265442929, 0.283576113255,-0.0842718450726,0.931086305879, -1.05296584292,0.432020532920,-0.251108254803, 0.127725582443,0.0483621161849,-0.0116473795607, 0.000334005754773,1.↩
37016097588,-2.12444673002, 0.578908942724,0.304945770499,-0.184276165165, -0.292111460397,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 3.0,4.0,5.0, 0.0,1.0,2.0, 3.0,4.0,0.0, 1.0,2.0,0.0, 1.0,0.0,1.0, 1.0,3.0,4.0, 5.0,3.0,4.0, 5.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 0,0,0,1,1,1, 1,1,2,2,2,3, 3,4,4,5,0,0, 0,2,2,2,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,2,2, 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 0.0d0,0.0d0,0.0d0,0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.0d0,0.0d0,0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_↩
na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 4, c_id = (/ 4.00007687959299,6.↩
096117164499717,6.262120355002826, 8.638166021481062,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,4380.0,1180.0, 1810.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -7.346431336306658, a2_id = 5.3030265687291829)

- type(meosdata), parameter **meos_66** = meosdata(ident = "PRLN", name = "propylen", default_ref_state = "IIR", bibref = "Unpublished. Lemmon, McLinden, Overhoff and Wagner (2018). A Reference Equation ", mw = 42.07974, tc = 364.211, pc = 4555.0, rhoc = 5.457, ttr = 87.953, ptr = 0.0000007471, t_nbp = 225.531, tr = 364.211, rhor = 5.457, Rgas = 8.314472, acf = 0.146, t_max = 575.0, p_max = 1000000.↩
0, n_poly_eos = 6, n_exp_eos = 6, n_gauss_eos = 9, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.04341002,1.136592,-0.8528611, 0.5216669,-1.382953,0.1214347, -0.5984662,-1.391883,-1.008434, 0.1961249,-0.360693,-0.002407175, 0.7432121,0.1475162,-0.02503391, -0.2734409,0.006378889,0.↩
0150294, -0.03162971,-0.04107194,-1.190241, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.205,0.56, 0.676,1.0,0.5, 1.0,1.94,2.0, 1.0,2.66,0.83, 1.6,2.5,3.0, 2.5,2.72,4.0, 4.0,1.0,4.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 4,1,1,2,2,3, 1,1,3,2,2,8, 1,1,2,3,3,2, 1,2,3,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 1,2,2,1,2,1, 2,2,2,2,2,2, 2,2,2,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.↩
0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -1.07,-0.66,-1.2,-1.12,-1.47,-1.93, -3.3,-15.4,-6.0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -0.77,-0.83,-0.607,-0.4,-0.66,-0.07, -3.1,-387.0,-41.0,0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0 /), gamma_eos = (/ 1.21,1.08,0.83,0.56,1.22,1.81, 1.54,1.12,1.4,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0

/), epsilon_eos = (/ 0.78,0.82,1.94,0.69,1.96,1.3, 0.38,0.91,0.7,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp↩
_eos = (/ 2,2,2,2,2,2, 2,2,2,0,0,0, 0,0 /), del_exp_eos = (/ 2,2,2,2,2,2, 2,2,2,0,0,0, 0,0 /), b_assoc_eos = (/
0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0
/), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0
/), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/
4.0,1.544,4.013, 8.923,6.02,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,324.0,973.0,
1932.0,4317.0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -5.1823353913028285, a2_id
= 4.3639943544853246)

- type(meosdata), parameter **meos_67** = meosdata(ident = "R41", name = "r41", default_ref_state = "IIR", bibref = "DOI: 10.1021/je050186n", mw = 34.03292, tc = 317.28, pc = 5897.0, rhoc = 9.3, ttr = 129.82, ptr = 0.345, t_nbp = 194.84, tr = 317.28, rhor = 9.3, Rgas = 8.314472, acf = 0.2004, t_max = 425.0, p↩
_max = 70000.0, n_poly_eos = 5, n_exp_eos = 7, n_gauss_eos = 0, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 1.6264,-2.8337,0.0010932, 0.037136,0.00018724,-0.22189, 0.55021,0.0461,-0.056405, -0.↩
17005,-0.032409,-0.012276, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 0.↩
52,1.12,4.0, 0.03,0.63,3.4, 2.2,1.5,0.1, 4.8,3.5,15.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), d_eos = (/ 1,1,1,3,7,1, 2,5,1,1,4,2, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,1, 1,1,2,2,2,3, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.↩
0d0, 1.0d0,1.0d0,1.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/
0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/
0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), del_exp_eos = (/ 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0
/), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na =
(/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/
0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 2, n_id = 4, c_id = (/ 4.0,0.00016937,5.↩
6936, 2.9351,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,1.0,1841.0, 4232.↩
0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = -4.8676501221130053, a2_id = 4.↩
2527989117950007)
- type(meosdata), parameter **meos_68** = meosdata(ident = "MXYL", name = "mxylene", default_ref_state = "NBP", bibref = "DOI: 10.1063/1.3703506", mw = 106.165, tc = 616.89, pc = 3534.6, rhoc = 2.665, ttr = 225.3, ptr = 0.003123, t_nbp = 412.214, tr = 616.89, rhor = 2.665, Rgas = 8.314472, acf = 0.326, t_↩
max = 700.0, p_max = 200000.0, n_poly_eos = 6, n_exp_eos = 5, n_gauss_eos = 4, n_nona_eos = 0, n_assoc_eos = 0, n_eos = (/ 0.000012791017,0.041063111,1.505996, -2.3095875,-0.46969,0.171031, -1.↩
001728,-0.3945766,0.6970578, -0.3002876,-0.024311,0.815488, -0.330647,-0.123393,-0.54661, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_eos = (/ 1.0,0.91,0.231, 0.772,1.205,0.323, 2.7,3.11,0.768, 4.1,0.818,2.0, 2.9,3.83,0.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_eos = (/ 8,4,1,1,2,3, 1,3,2,2,7,1, 1,3,3,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), l_eos = (/ 0,0,0,0,0,0, 2,2,1,2,1,2, 2,2,2,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0 /), g_eos = (/ 1.0d0,1.↩
0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,1.0d0, 1.0d0,1.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0, 0.0d0,0.0d0,0.0d0 /), eta_eos = (/ -1.0244,-1.3788,-0.9806,-6.3563,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0,0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), beta_eos = (/ -1.66,-1.9354,-1.0323,-78.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), gamma_eos = (/ 1.1013,0.6515,0.4975,1.26,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), epsilon_eos = (/ 0.713,0.9169,0.6897,0.7245,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0 /), tau_exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), del_↩

exp_eos = (/ 2,2,2,2,0,0, 0,0,0,0,0,0, 0,0 /), b_assoc_eos = (/ 0.0d0,0.0d0 /), n_na = (/ 0.0d0,0.0d0,0.0d0 /), a_na = (/ 0.0d0,0.0d0,0.0d0 /), b_na = (/ 0.0d0,0.0d0,0.0d0 /), beta_na = (/ 0.0d0,0.0d0,0.0d0 /), big_a_na = (/ 0.0d0,0.0d0,0.0d0 /), big_b_na = (/ 0.0d0,0.0d0,0.0d0 /), big_c_na = (/ 0.0d0,0.0d0,0.0d0 /), big_d_na = (/ 0.0d0,0.0d0,0.0d0 /), n1_id = 1, n_id = 5, c_id = (/ 2.169909,4.44312,2.862794, 24.83298,16.26077,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), t_id = (/ 0.0,160.0,190.0, 1333.0,3496.0,0.0d0, 0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0 /), a1_id = 12.6528905042467272, a2_id = -0.4597586328446329)

- integer, parameter **maxmeos** = 68
- type(meosdata), dimension(maxmeos), parameter **meosdb** = (/ meos_1,meos_2,meos_3,meos_4,meos_5,meos_6, meos_7,meos_8,meos_9,meos_10,meos_11,meos_12, meos_13,meos_14,meos_15,meos_16,meos_17,meos_18, meos_19,meos_20,meos_21,meos_22,meos_23,meos_24, meos_25,meos_26,meos_27,meos_28,meos_29,meos_30, meos_31,meos_32,meos_33,meos_34,meos_35,meos_36, meos_37,meos_38,meos_39,meos_40,meos_41,meos_42, meos_43,meos_44,meos_45,meos_46,meos_47,meos_48, meos_49,meos_50,meos_51,meos_52,meos_53,meos_54, meos_55,meos_56,meos_57,meos_58,meos_59,meos_60, meos_61,meos_62,meos_63,meos_64,meos_65,meos_66, meos_67,meos_68 /)

5.32.1 Detailed Description

Automatically generated file meosdatadb.f90 Time stamp: 2023-03-23T14:18:36.228460.

5.33 meosmixdb Module Reference

Automatically generated file meosmixdb.f90 Time stamp: 2023-03-23T14:50:21.734496.

Data Types

- type meos_mix_data
- type meos_mix_reducing

Variables

- type(meos_mix_reducing), parameter **meos_red_1** = meos_mix_reducing(ident1 = "N2", ident2 = "NH3", bibref = "Neumann et al. (2020)", beta_v = 0.739937, gamma_v = 1.447261, beta_T = 1.057512, gamma_T = 0.952705)
- type(meos_mix_reducing), parameter **meos_red_2** = meos_mix_reducing(ident1 = "H2", ident2 = "NH3", bibref = "Neumann et al. (2020)", beta_v = 1.0103, gamma_v = 0.7298, beta_T = 0.98824, gamma_T = 1.1266)
- type(meos_mix_reducing), parameter **meos_red_3** = meos_mix_reducing(ident1 = "N2", ident2 = "BENZENE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.435)
- type(meos_mix_reducing), parameter **meos_red_4** = meos_mix_reducing(ident1 = "CO2", ident2 = "H2", bibref = "Beckmueller et al. (2020) ", beta_v = 1.198000, gamma_v = 0.842000, beta_T = 0.979000, gamma_T = 1.961000)
- type(meos_mix_reducing), parameter **meos_red_5** = meos_mix_reducing(ident1 = "R115", ident2 = "R114", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0021192, beta_T = 1., gamma_T = 1.0008497)
- type(meos_mix_reducing), parameter **meos_red_6** = meos_mix_reducing(ident1 = "NC10", ident2 = "O2", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_7** = meos_mix_reducing(ident1 = "C1", ident2 = "ETOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.954, gamma_T = 1.366)
- type(meos_mix_reducing), parameter **meos_red_8** = meos_mix_reducing(ident1 = "C1", ident2 = "H2S", bibref = "Kunz and Wagner (2012)", beta_v = 1.012599087, gamma_v = 1.040161207, beta_T = 1.011090031, gamma_T = 0.961155729)
- type(meos_mix_reducing), parameter **meos_red_9** = meos_mix_reducing(ident1 = "AR", ident2 = "BENZENE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.01, gamma_T = 1.397)

- type(meos_mix_reducing), parameter **meos_red_10** = meos_mix_reducing(ident1 = "R114", ident2 = "NC6", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.2% from 12 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.003723815355, gamma_T = 0.98179)
- type(meos_mix_reducing), parameter **meos_red_11** = meos_mix_reducing(ident1 = "C2", ident2 = "PRLN", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.002)
- type(meos_mix_reducing), parameter **meos_red_12** = meos_mix_reducing(ident1 = "R23", ident2 = "IC5", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 3.02% from 26 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.029261916280, gamma_T = 0.93001)
- type(meos_mix_reducing), parameter **meos_red_13** = meos_mix_reducing(ident1 = "NC4", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.02)
- type(meos_mix_reducing), parameter **meos_red_14** = meos_mix_reducing(ident1 = "MEOH", ident2 = "NC7", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.0533466, beta_T = 1., gamma_T = 0.8586313)
- type(meos_mix_reducing), parameter **meos_red_15** = meos_mix_reducing(ident1 = "R23", ident2 = "C3", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.77% from 32 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.012002347845, gamma_T = 0.87585)
- type(meos_mix_reducing), parameter **meos_red_16** = meos_mix_reducing(ident1 = "AR", ident2 = "NC10", bibref = "Mixing rules of nitrogen/decane used as an estimate.", beta_v = 1., gamma_v = 1., beta_T = 0.↩ 957934447, gamma_T = 1.822157123)
- type(meos_mix_reducing), parameter **meos_red_17** = meos_mix_reducing(ident1 = "R114", ident2 = "NC5", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.12% from 12 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.003552576119, gamma_T = 0.97011)
- type(meos_mix_reducing), parameter **meos_red_18** = meos_mix_reducing(ident1 = "BENZENE", ident2 = "MXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.82% from 53 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.99896, gamma_T = 1.00464)
- type(meos_mix_reducing), parameter **meos_red_19** = meos_mix_reducing(ident1 = "N2", ident2 = "NC6", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.195952177, beta_T = 1., gamma_T = 1.↩ 472607971)
- type(meos_mix_reducing), parameter **meos_red_20** = meos_mix_reducing(ident1 = "C2", ident2 = "C3", bibref = "Kunz and Wagner (2007)", beta_v = 0.997607277, gamma_v = 1.00303472, beta_T = 0.996199694, gamma_T = 1.01473019)
- type(meos_mix_reducing), parameter **meos_red_21** = meos_mix_reducing(ident1 = "TOLU", ident2 = "NC9", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010) (estimated from trend found in C1-C10)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_22** = meos_mix_reducing(ident1 = "CO2", ident2 = "NC9", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 0.973386152, beta_T = 1.00768862, gamma_T = 1.140671202)
- type(meos_mix_reducing), parameter **meos_red_23** = meos_mix_reducing(ident1 = "NC7", ident2 = "NC8", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.006767176, beta_T = 1., gamma_T = 0.↩ 998793111)
- type(meos_mix_reducing), parameter **meos_red_24** = meos_mix_reducing(ident1 = "NC8", ident2 = "MXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.56% from 24 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.9996901, gamma_T = 0.99296)
- type(meos_mix_reducing), parameter **meos_red_25** = meos_mix_reducing(ident1 = "NC5", ident2 = "MEOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 0.6340970, beta_T = 1., gamma_T = 0.9168380)
- type(meos_mix_reducing), parameter **meos_red_26** = meos_mix_reducing(ident1 = "NC6", ident2 = "NC10", bibref = "Kunz and Wagner (2012)", beta_v = 1.001516371, gamma_v = 1.013511439, beta_T = 0.99764101, gamma_T = 1.028939539)
- type(meos_mix_reducing), parameter **meos_red_27** = meos_mix_reducing(ident1 = "R32", ident2 = "PRLN", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.36% from 80 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.003673444808, gamma_T = 0.90627)
- type(meos_mix_reducing), parameter **meos_red_28** = meos_mix_reducing(ident1 = "NH3", ident2 = "NC5", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.↩ 161980013944, gamma_T = 0.8702)

- type(meos_mix_reducing), parameter **meos_red_29** = meos_mix_reducing(ident1 = "CO2", ident2 = "R1234YF", bibref = "E.W. Lemmon, NIST (2013); based on simulation data from Gabriele Raabe", beta_v = 1., gamma_v = 1.015, beta_T = 1.017, gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_30** = meos_mix_reducing(ident1 = "NC4", ident2 = "NC12", bibref = "J. Watanasiri and E.W. Lemmon, NIST (2010) (estimated from propane and pentane mixed with dodecane)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.15)
- type(meos_mix_reducing), parameter **meos_red_31** = meos_mix_reducing(ident1 = "N2", ident2 = "PRLN", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.99, gamma_T = 1.197)
- type(meos_mix_reducing), parameter **meos_red_32** = meos_mix_reducing(ident1 = "NH3", ident2 = "CYCLOHEX", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.3341, beta_T = 1.107297087809, gamma_T = 0.8395)
- type(meos_mix_reducing), parameter **meos_red_33** = meos_mix_reducing(ident1 = "NC4", ident2 = "NC7", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.019174227, beta_T = 1., gamma_T = 1.↩ 021283378)
- type(meos_mix_reducing), parameter **meos_red_34** = meos_mix_reducing(ident1 = "NC10", ident2 = "NC12", bibref = "J. Watanasiri and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.005, beta_T = 1., gamma_T = 0.998)
- type(meos_mix_reducing), parameter **meos_red_35** = meos_mix_reducing(ident1 = "CO2", ident2 = "NC10", bibref = "Kunz and Wagner (2012)", beta_v = 1.000151132, gamma_v = 1.183394668, beta_T = 1.02002879, gamma_T = 1.145512213)
- type(meos_mix_reducing), parameter **meos_red_36** = meos_mix_reducing(ident1 = "C3", ident2 = "NC12", bibref = "J. Watanasiri and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.19)
- type(meos_mix_reducing), parameter **meos_red_37** = meos_mix_reducing(ident1 = "R23", ident2 = "R32", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.36% from 16 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.00116, gamma_T = 1.00385)
- type(meos_mix_reducing), parameter **meos_red_38** = meos_mix_reducing(ident1 = "C2", ident2 = "IC4", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.006616886, beta_T = 1., gamma_T = 1.↩ 033283811)
- type(meos_mix_reducing), parameter **meos_red_39** = meos_mix_reducing(ident1 = "NE", ident2 = "XE", bibref = "E.W. Lemmon, NIST (2017)", beta_v = 0.894, gamma_v = 1.213, beta_T = 1., gamma_T = 1.283)
- type(meos_mix_reducing), parameter **meos_red_40** = meos_mix_reducing(ident1 = "OXYL", ident2 = "NC9", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.15% from 40 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.002908434460, gamma_T = 0.98501)
- type(meos_mix_reducing), parameter **meos_red_41** = meos_mix_reducing(ident1 = "ETOH", ident2 = "CYCLOHEX", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 0.994035785288, gamma_v = 1.2, beta_T = 0.989119683482, gamma_T = 0.961)
- type(meos_mix_reducing), parameter **meos_red_42** = meos_mix_reducing(ident1 = "ETOH", ident2 = "NC9", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.048, beta↩ _T = 1.021450459653, gamma_T = 0.9)
- type(meos_mix_reducing), parameter **meos_red_43** = meos_mix_reducing(ident1 = "CO", ident2 = "H2", bibref = "Beckmueller et al. (2020)", beta_v = 1.037000, gamma_v = 1.040000, beta_T = 1.078000, gamma↩ _T = 1.105000)
- type(meos_mix_reducing), parameter **meos_red_44** = meos_mix_reducing(ident1 = "C2", ident2 = "MXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 2.32% from 34 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.98799, gamma_T = 1.17616)
- type(meos_mix_reducing), parameter **meos_red_45** = meos_mix_reducing(ident1 = "SO2", ident2 = "CO", bibref = "Herrig (2018) - Linear Combining Rules / see Herrig (2018) PhD thesis", beta_v = 1.000000000, gamma_v = 1.007241430, beta_T = 1.000000000, gamma_T = 1.177903242)
- type(meos_mix_reducing), parameter **meos_red_46** = meos_mix_reducing(ident1 = "IC4", ident2 = "ACETONE", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.↩ 016363451570, gamma_T = 0.9358)
- type(meos_mix_reducing), parameter **meos_red_47** = meos_mix_reducing(ident1 = "C2", ident2 = "NC6", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.169701102, beta_T = 1., gamma_T = 1.↩ 092177796)

- type(meos_mix_reducing), parameter **meos_red_48** = meos_mix_reducing(ident1 = "N2", ident2 = "ETOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.927, gamma_T = 1.666)
- type(meos_mix_reducing), parameter **meos_red_49** = meos_mix_reducing(ident1 = "R12", ident2 = "NC7", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.43% from 4 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.011920422578, gamma_T = 1.03388)
- type(meos_mix_reducing), parameter **meos_red_50** = meos_mix_reducing(ident1 = "NC4", ident2 = "CYCLOHEX", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.0267)
- type(meos_mix_reducing), parameter **meos_red_51** = meos_mix_reducing(ident1 = "MEOH", ident2 = "H2O", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.034, beta_T = 0.995024875622, gamma_T = 0.961)
- type(meos_mix_reducing), parameter **meos_red_52** = meos_mix_reducing(ident1 = "AR", ident2 = "ETOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.492)
- type(meos_mix_reducing), parameter **meos_red_53** = meos_mix_reducing(ident1 = "NC6", ident2 = "MXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.42% from 11 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.01308, gamma_T = 0.99672)
- type(meos_mix_reducing), parameter **meos_red_54** = meos_mix_reducing(ident1 = "IC4", ident2 = "IC5", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.002284353, beta_T = 1., gamma_T = 1.001835788)
- type(meos_mix_reducing), parameter **meos_red_55** = meos_mix_reducing(ident1 = "MEOH", ident2 = "BENZENE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.978473581213, gamma_T = 0.901)
- type(meos_mix_reducing), parameter **meos_red_56** = meos_mix_reducing(ident1 = "C3", ident2 = "NC10", bibref = "Kunz and Wagner (2012)", beta_v = 0.984104227, gamma_v = 1.053040574, beta_T = 0.985331233, gamma_T = 1.140905252)
- type(meos_mix_reducing), parameter **meos_red_57** = meos_mix_reducing(ident1 = "XE", ident2 = "C3_1", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 3.05% from 15 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.934273835661, gamma_T = 0.94711)
- type(meos_mix_reducing), parameter **meos_red_58** = meos_mix_reducing(ident1 = "H2", ident2 = "H2S", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_59** = meos_mix_reducing(ident1 = "IC4", ident2 = "ETOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.018, gamma_T = 0.965)
- type(meos_mix_reducing), parameter **meos_red_60** = meos_mix_reducing(ident1 = "C1", ident2 = "O2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 0.95)
- type(meos_mix_reducing), parameter **meos_red_61** = meos_mix_reducing(ident1 = "NH3", ident2 = "O2", bibref = "Neumann et al. (2020)", beta_v = 1.000000, gamma_v = 1.000002, beta_T = 1.000000, gamma_T = 1.118566)
- type(meos_mix_reducing), parameter **meos_red_62** = meos_mix_reducing(ident1 = "ACETONE", ident2 = "MEG", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 0.9108, beta_T = 0.9975, gamma_T = 1.0468)
- type(meos_mix_reducing), parameter **meos_red_63** = meos_mix_reducing(ident1 = "CO2", ident2 = "R41", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 0.9771864, beta_T = 1., gamma_T = 0.9941317)
- type(meos_mix_reducing), parameter **meos_red_64** = meos_mix_reducing(ident1 = "CO2", ident2 = "BENZENE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1.044, gamma_v = 1.152, beta_T = 0.99, gamma_T = 0.966)
- type(meos_mix_reducing), parameter **meos_red_65** = meos_mix_reducing(ident1 = "IC4", ident2 = "NC12", bibref = "J. Watanasiri and E.W. Lemmon, NIST (2010) (estimated from propane and pentane mixed with dodecane)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.16)
- type(meos_mix_reducing), parameter **meos_red_66** = meos_mix_reducing(ident1 = "KR", ident2 = "C2", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.992161920825, gamma_T = 1.0437)
- type(meos_mix_reducing), parameter **meos_red_67** = meos_mix_reducing(ident1 = "R23", ident2 = "NC5", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 3.01% from 26 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.034960981971, gamma_T = 0.94007)

- type(meos_mix_reducing), parameter **meos_red_68** = meos_mix_reducing(ident1 = "ACETONE", ident2 = "CYCLOHEX", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 0.9903, beta_T = 1., gamma_T = 0.9335)
- type(meos_mix_reducing), parameter **meos_red_69** = meos_mix_reducing(ident1 = "C1", ident2 = "NC4", bibref = "Kunz and Wagner (2007)", beta_v = 0.979105972, gamma_v = 1.045375122, beta_T = 0.99417491, gamma_T = 1.171607691)
- type(meos_mix_reducing), parameter **meos_red_70** = meos_mix_reducing(ident1 = "ETOH", ident2 = "NC8", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 0.986193293886, gamma_v = 1.036, beta_T = 0.985221674877, gamma_T = 0.887)
- type(meos_mix_reducing), parameter **meos_red_71** = meos_mix_reducing(ident1 = "CO", ident2 = "HE", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_72** = meos_mix_reducing(ident1 = "SO2", ident2 = "ACETONE", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.990197049213, gamma_T = 1.0959)
- type(meos_mix_reducing), parameter **meos_red_73** = meos_mix_reducing(ident1 = "XE", ident2 = "NC6", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.81% from 21 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.980786394531, gamma_T = 1.16656)
- type(meos_mix_reducing), parameter **meos_red_74** = meos_mix_reducing(ident1 = "PRLN", ident2 = "H2O", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.931, gamma_T = 0.8775)
- type(meos_mix_reducing), parameter **meos_red_75** = meos_mix_reducing(ident1 = "R134A", ident2 = "MEOH", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 3.08% from 14 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.992270215025, gamma_T = 0.98889)
- type(meos_mix_reducing), parameter **meos_red_76** = meos_mix_reducing(ident1 = "MEOH", ident2 = "NC12", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.6% from 52 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.109742428782, gamma_T = 0.79118)
- type(meos_mix_reducing), parameter **meos_red_77** = meos_mix_reducing(ident1 = "R14", ident2 = "C2", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 1.39% from 29 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.985386715016, gamma_T = 0.89216)
- type(meos_mix_reducing), parameter **meos_red_78** = meos_mix_reducing(ident1 = "CO2", ident2 = "PRLN", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1.096, gamma_v = 1.014, beta_T = 0.997, gamma_T = 0.945)
- type(meos_mix_reducing), parameter **meos_red_79** = meos_mix_reducing(ident1 = "R23", ident2 = "R143A", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.72% from 16 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.008095, gamma_T = 1.00381)
- type(meos_mix_reducing), parameter **meos_red_80** = meos_mix_reducing(ident1 = "IC4", ident2 = "NC10", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.060243344, beta_T = 1., gamma_T = 1.021624748)
- type(meos_mix_reducing), parameter **meos_red_81** = meos_mix_reducing(ident1 = "R32", ident2 = "IC4", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0454476, beta_T = 1., gamma_T = 0.8573096)
- type(meos_mix_reducing), parameter **meos_red_82** = meos_mix_reducing(ident1 = "NH3", ident2 = "SO2", bibref = "Neumann et al. (2020)", beta_v = 1.000000, gamma_v = 1.000000, beta_T = 1.000000, gamma_T = 1.000000)
- type(meos_mix_reducing), parameter **meos_red_83** = meos_mix_reducing(ident1 = "R23", ident2 = "R116", bibref = "E.W. Lemmon, NIST (2004)", beta_v = 1., gamma_v = 1.0229397, beta_T = 1., gamma_T = 0.9090532)
- type(meos_mix_reducing), parameter **meos_red_84** = meos_mix_reducing(ident1 = "F6S", ident2 = "C3", bibref = "E.W. Lemmon, NIST (2008)", beta_v = 1.007559212995, gamma_v = 1.0150842, beta_T = 0.990884850262, gamma_T = 0.89338144)
- type(meos_mix_reducing), parameter **meos_red_85** = meos_mix_reducing(ident1 = "KR", ident2 = "PRLN", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0390195, beta_T = 1., gamma_T = 1.0773977)
- type(meos_mix_reducing), parameter **meos_red_86** = meos_mix_reducing(ident1 = "C1", ident2 = "N2", bibref = "Kunz and Wagner (2007)", beta_v = 0.998721377, gamma_v = 1.013950311, beta_T = 0.99809883, gamma_T = 0.979273013)

- type(meos_mix_reducing), parameter **meos_red_87** = meos_mix_reducing(ident1 = "CO2", ident2 = "NC6", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 0.851343711, beta_T = 1., gamma_T = 1.↩ 038675574)
- type(meos_mix_reducing), parameter **meos_red_88** = meos_mix_reducing(ident1 = "NC4", ident2 = "IC4", bibref = "Kunz and Wagner (2007)", beta_v = 1.000880464, gamma_v = 1.00041444, beta_T = 1.000077547, gamma_T = 1.001432824)
- type(meos_mix_reducing), parameter **meos_red_89** = meos_mix_reducing(ident1 = "NC6", ident2 = "NC8", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.006268954, beta_T = 1., gamma_T = 1.↩ 001633952)
- type(meos_mix_reducing), parameter **meos_red_90** = meos_mix_reducing(ident1 = "C2", ident2 = "NC7", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.057666085, beta_T = 1., gamma_T = 1.↩ 134532014)
- type(meos_mix_reducing), parameter **meos_red_91** = meos_mix_reducing(ident1 = "C3", ident2 = "AR", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_92** = meos_mix_reducing(ident1 = "NC10", ident2 = "H2↩ S", bibref = "Kunz and Wagner (2012)", beta_v = 0.975187766, gamma_v = 1.171714677, beta_T = 0.↩ 973091413, gamma_T = 1.103693489)
- type(meos_mix_reducing), parameter **meos_red_93** = meos_mix_reducing(ident1 = "PRLN", ident2 = "BEN-ZENE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.03)
- type(meos_mix_reducing), parameter **meos_red_94** = meos_mix_reducing(ident1 = "NC5", ident2 = "NC6", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.002480637, beta_T = 1., gamma_T = 1.↩ 000761237)
- type(meos_mix_reducing), parameter **meos_red_95** = meos_mix_reducing(ident1 = "R32", ident2 = "SO2", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.2% from 52 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.002379950717, gamma_T = 0.99968)
- type(meos_mix_reducing), parameter **meos_red_96** = meos_mix_reducing(ident1 = "R14", ident2 = "R23", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0002627, beta_T = 1., gamma_T = 0.↩ 9467580)
- type(meos_mix_reducing), parameter **meos_red_97** = meos_mix_reducing(ident1 = "PRLN", ident2 = "NC4", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010) (estimated from trend found in C1-C10)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.02)
- type(meos_mix_reducing), parameter **meos_red_98** = meos_mix_reducing(ident1 = "MEOH", ident2 = "NC8", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.124, beta↩ _T = 1.074113856069, gamma_T = 0.881)
- type(meos_mix_reducing), parameter **meos_red_99** = meos_mix_reducing(ident1 = "NC8", ident2 = "OXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.84% from 36 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.99933, gamma_T = 0.99245)
- type(meos_mix_reducing), parameter **meos_red_100** = meos_mix_reducing(ident1 = "C3", ident2 = "H2↩ O", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.011759763, beta_T = 1., gamma_T = 0.600340961)
- type(meos_mix_reducing), parameter **meos_red_101** = meos_mix_reducing(ident1 = "CO2", ident2 = "R143A", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.63% from 61 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.02027, gamma_T = 0.98474)
- type(meos_mix_reducing), parameter **meos_red_102** = meos_mix_reducing(ident1 = "R143A", ident2 = "R12", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0007214, beta_T = 1., gamma_T = 0.9484400)
- type(meos_mix_reducing), parameter **meos_red_103** = meos_mix_reducing(ident1 = "PRLN", ident2 = "MXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 2.3% from 16 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 0.9934926, gamma_T = 1.09811)
- type(meos_mix_reducing), parameter **meos_red_104** = meos_mix_reducing(ident1 = "R23", ident2 = "CO2", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0099333, beta_T = 1., gamma_T = 0.↩ 9797140)
- type(meos_mix_reducing), parameter **meos_red_105** = meos_mix_reducing(ident1 = "NC6", ident2 = "H2↩ O", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.170217596, beta_T = 1., gamma_T = 0.569681333)

- type(meos_mix_reducing), parameter **meos_red_106** = meos_mix_reducing(ident1 = "NC4", ident2 = "IC5", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.002728434, beta_T = 1., gamma_T = 1.↩ 000792201)
- type(meos_mix_reducing), parameter **meos_red_107** = meos_mix_reducing(ident1 = "MXYL", ident2 = "NC11", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.1% from 13 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 0.999100809272, gamma_T = 1.00198)
- type(meos_mix_reducing), parameter **meos_red_108** = meos_mix_reducing(ident1 = "CO", ident2 = "PRLN", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010) (estimated from nitrogen and argon)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.184)
- type(meos_mix_reducing), parameter **meos_red_109** = meos_mix_reducing(ident1 = "XE", ident2 = "R41", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 2.08% from 31 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.0229865, gamma_T = 0.91395)
- type(meos_mix_reducing), parameter **meos_red_110** = meos_mix_reducing(ident1 = "CO2", ident2 = "PXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 2.98% from 119 bubble-poi", beta_v = 1., gamma_v = 1., beta_T = 1.0034, gamma_T = 1.1043)
- type(meos_mix_reducing), parameter **meos_red_111** = meos_mix_reducing(ident1 = "NC4", ident2 = "H2↩ O", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.223638763, beta_T = 1., gamma_T = 0.615512682)
- type(meos_mix_reducing), parameter **meos_red_112** = meos_mix_reducing(ident1 = "CYCLOHEX", ident2 = "NC12", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.0167, beta_T = 1., gamma_T = 1.0548)
- type(meos_mix_reducing), parameter **meos_red_113** = meos_mix_reducing(ident1 = "AR", ident2 = "NH3", bibref = "Neumann et al. (2020)", beta_v = 0.756526, gamma_v = 1.041113, beta_T = 1.146326, gamma_T = 0.998353)
- type(meos_mix_reducing), parameter **meos_red_114** = meos_mix_reducing(ident1 = "ACETONE", ident2 = "NC6", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 0.9194)
- type(meos_mix_reducing), parameter **meos_red_115** = meos_mix_reducing(ident1 = "IC5", ident2 = "BEN-ZENE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010) (estimated from trend found in C1-C10)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 0.99)
- type(meos_mix_reducing), parameter **meos_red_116** = meos_mix_reducing(ident1 = "F6S", ident2 = "R32", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.48% from 12 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.0017029, gamma_T = 0.86502)
- type(meos_mix_reducing), parameter **meos_red_117** = meos_mix_reducing(ident1 = "CO2", ident2 = "R116", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.28% from 77 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.03808, gamma_T = 0.89857)
- type(meos_mix_reducing), parameter **meos_red_118** = meos_mix_reducing(ident1 = "N2", ident2 = "C2", bibref = "Kunz and Wagner (2007)", beta_v = 0.978880168, gamma_v = 1.042352891, beta_T = 1.↩ 007671428, gamma_T = 1.098650964)
- type(meos_mix_reducing), parameter **meos_red_119** = meos_mix_reducing(ident1 = "R32", ident2 = "MEOH", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 3.41% from 17 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.007049345418, gamma_T = 1.01175)
- type(meos_mix_reducing), parameter **meos_red_120** = meos_mix_reducing(ident1 = "IC5", ident2 = "H2↩ S", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 0.835763343, beta_T = 1., gamma_T = 0.982651529)
- type(meos_mix_reducing), parameter **meos_red_121** = meos_mix_reducing(ident1 = "NC10", ident2 = "H2O", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 0.551405318, beta_T = 0.897162268, gamma_T = 0.740416402)
- type(meos_mix_reducing), parameter **meos_red_122** = meos_mix_reducing(ident1 = "NC6", ident2 = "NC7", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.001508227, beta_T = 1., gamma_T = 0.↩ 999762786)
- type(meos_mix_reducing), parameter **meos_red_123** = meos_mix_reducing(ident1 = "ETOH", ident2 = "PXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 3.86% from 190 bubble-poi", beta_v = 1., gamma_v = 1., beta_T = 0.97813, gamma_T = 0.9059)
- type(meos_mix_reducing), parameter **meos_red_124** = meos_mix_reducing(ident1 = "C2", ident2 = "R116", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.26% from 60 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.03539, gamma_T = 0.89844)

- type(meos_mix_reducing), parameter **meos_red_125** = meos_mix_reducing(ident1 = "IC4", ident2 = "NC7", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.021668316, beta_T = 1., gamma_T = 1.↩ 00988576)
- type(meos_mix_reducing), parameter **meos_red_126** = meos_mix_reducing(ident1 = "C3", ident2 = "R114", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.17% from 12 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.00332, gamma_T = 0.98055)
- type(meos_mix_reducing), parameter **meos_red_127** = meos_mix_reducing(ident1 = "CYCLOHEX", ident2 = "NC9", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010) (estimated from octane/cyclohexane and dodecane/cyclohexan", beta_v = 1., gamma_v = 1.009, beta_T = 1., gamma_T = 1.015)
- type(meos_mix_reducing), parameter **meos_red_128** = meos_mix_reducing(ident1 = "BENZENE", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.004, beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_129** = meos_mix_reducing(ident1 = "NC9", ident2 = "O2", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_130** = meos_mix_reducing(ident1 = "C1", ident2 = "NC8", bibref = "Kunz and Wagner (2007)", beta_v = 0.994740603, gamma_v = 1.116549372, beta_T = 0.↩ 957473785, gamma_T = 1.449245409)
- type(meos_mix_reducing), parameter **meos_red_131** = meos_mix_reducing(ident1 = "PRLN", ident2 = "NC10", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.956937799043, gamma_T = 1.191)
- type(meos_mix_reducing), parameter **meos_red_132** = meos_mix_reducing(ident1 = "R12", ident2 = "R114", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0085162, beta_T = 1., gamma_T = 0.9983405)
- type(meos_mix_reducing), parameter **meos_red_133** = meos_mix_reducing(ident1 = "NC9", ident2 = "HE", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_134** = meos_mix_reducing(ident1 = "NC9", ident2 = "H2", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.342647661, beta_T = 1., gamma_T = 2.↩ 23435404)
- type(meos_mix_reducing), parameter **meos_red_135** = meos_mix_reducing(ident1 = "NC4", ident2 = "CO", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.084740904, beta_T = 1., gamma_T = 1.↩ 173916162)
- type(meos_mix_reducing), parameter **meos_red_136** = meos_mix_reducing(ident1 = "O2", ident2 = "BENZENE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010) (estimated from argon/benzene)", beta_v = 1., gamma_v = 1., beta_T = 1.01, gamma_T = 1.397)
- type(meos_mix_reducing), parameter **meos_red_137** = meos_mix_reducing(ident1 = "R114", ident2 = "ETOH", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 3.↩ 63% from 6 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.048646721406, gamma_T = 0.93497)
- type(meos_mix_reducing), parameter **meos_red_138** = meos_mix_reducing(ident1 = "NC9", ident2 = "H2O", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_139** = meos_mix_reducing(ident1 = "NC7", ident2 = "TOLU", bibref = "E.W. Lemmon, NIST (2012)", beta_v = 1.0019167, gamma_v = 1.0026789, beta_T = 0.998329, gamma_T = 0.9832346)
- type(meos_mix_reducing), parameter **meos_red_140** = meos_mix_reducing(ident1 = "NC4", ident2 = "NC8", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.046905515, beta_T = 1., gamma_T = 1.↩ 033180106)
- type(meos_mix_reducing), parameter **meos_red_141** = meos_mix_reducing(ident1 = "O2", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.45)
- type(meos_mix_reducing), parameter **meos_red_142** = meos_mix_reducing(ident1 = "PRLN", ident2 = "R134A", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0007007, beta_T = 1., gamma_T = 0.9364605)
- type(meos_mix_reducing), parameter **meos_red_143** = meos_mix_reducing(ident1 = "C2", ident2 = "NC4", bibref = "Kunz and Wagner (2007)", beta_v = 0.999157205, gamma_v = 1.006179146, beta_T = 0.↩ 999130554, gamma_T = 1.034832749)
- type(meos_mix_reducing), parameter **meos_red_144** = meos_mix_reducing(ident1 = "CYCLOHEX", ident2 = "EBZN", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.99% from 51 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.0043, gamma_T = 0.99414)

- type(meos_mix_reducing), parameter **meos_red_145** = meos_mix_reducing(ident1 = "N2", ident2 = "NC12", bibref = "J. Watanasiri and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.927, gamma_T = 1.949)
- type(meos_mix_reducing), parameter **meos_red_146** = meos_mix_reducing(ident1 = "NC5", ident2 = "NC16", bibref = "I.H. Bell, NIST (2017)", beta_v = 1., gamma_v = 1., beta_T = 0.9833, gamma_T = 1.1486)
- type(meos_mix_reducing), parameter **meos_red_147** = meos_mix_reducing(ident1 = "R114", ident2 = "R21", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0139360, beta_T = 1., gamma_T = 0.9609090)
- type(meos_mix_reducing), parameter **meos_red_148** = meos_mix_reducing(ident1 = "C3", ident2 = "NH3", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.9599, gamma_T = 0.7905)
- type(meos_mix_reducing), parameter **meos_red_149** = meos_mix_reducing(ident1 = "NC10", ident2 = "H2", bibref = "Kunz and Wagner (2012)", beta_v = 1.695358382, gamma_v = 1.120233729, beta_T = 1.↩ 064818089, gamma_T = 3.786003724)
- type(meos_mix_reducing), parameter **meos_red_150** = meos_mix_reducing(ident1 = "NC5", ident2 = "BENZENE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 0.976)
- type(meos_mix_reducing), parameter **meos_red_151** = meos_mix_reducing(ident1 = "NC7", ident2 = "H2↩ S", bibref = "Kunz and Wagner (2012)", beta_v = 0.828967164, gamma_v = 1.087956749, beta_T = 0.↩ 988937417, gamma_T = 1.013453092)
- type(meos_mix_reducing), parameter **meos_red_152** = meos_mix_reducing(ident1 = "R32", ident2 = "R143A", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0178106, beta_T = 1., gamma_T = 0.9748252)
- type(meos_mix_reducing), parameter **meos_red_153** = meos_mix_reducing(ident1 = "IC4", ident2 = "H2↩ S", bibref = "Kunz and Wagner (2012)", beta_v = 1.012994431, gamma_v = 0.988591117, beta_T = 0.↩ 974550548, gamma_T = 0.937130844)
- type(meos_mix_reducing), parameter **meos_red_154** = meos_mix_reducing(ident1 = "CO2", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1.047, gamma_v = 1.134, beta_T = 1., gamma_T = 1.024)
- type(meos_mix_reducing), parameter **meos_red_155** = meos_mix_reducing(ident1 = "IC5", ident2 = "ETOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.016, gamma_T = 0.934)
- type(meos_mix_reducing), parameter **meos_red_156** = meos_mix_reducing(ident1 = "ACETONE", ident2 = "NC7", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.0198, beta_T = 1., gamma_T = 0.9146)
- type(meos_mix_reducing), parameter **meos_red_157** = meos_mix_reducing(ident1 = "H2O", ident2 = "MEG", bibref = "I.H. Bell and E.W. Lemmon, NIST (2018)", beta_v = 1.009003234965, gamma_v = 1.↩ 066166, beta_T = 1.006062026742, gamma_T = 0.9986637)
- type(meos_mix_reducing), parameter **meos_red_158** = meos_mix_reducing(ident1 = "NC7", ident2 = "AR", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_159** = meos_mix_reducing(ident1 = "TOLU", ident2 = "NC8", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 0.996)
- type(meos_mix_reducing), parameter **meos_red_160** = meos_mix_reducing(ident1 = "N2", ident2 = "NC5", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.078877166, beta_T = 1., gamma_T = 1.↩ 419029041)
- type(meos_mix_reducing), parameter **meos_red_161** = meos_mix_reducing(ident1 = "NH3", ident2 = "MXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 3.7% from 138 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.042500015116, gamma_T = 0.95719)
- type(meos_mix_reducing), parameter **meos_red_162** = meos_mix_reducing(ident1 = "N2O", ident2 = "NC12", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 4.99% from 58 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.965763677628, gamma_T = 1.2736)
- type(meos_mix_reducing), parameter **meos_red_163** = meos_mix_reducing(ident1 = "NC5", ident2 = "NC7", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.008972412, beta_T = 1., gamma_T = 1.↩ 002441051)
- type(meos_mix_reducing), parameter **meos_red_164** = meos_mix_reducing(ident1 = "CO2", ident2 = "H2↩ S", bibref = "Kunz and Wagner (2012)", beta_v = 0.906630564, gamma_v = 1.024085837, beta_T = 1.↩ 016034583, gamma_T = 0.92601888)

- type(meos_mix_reducing), parameter **meos_red_165** = meos_mix_reducing(ident1 = "CO", ident2 = "ETOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.56)
- type(meos_mix_reducing), parameter **meos_red_166** = meos_mix_reducing(ident1 = "NC6", ident2 = "H2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.243461678, beta_T = 1., gamma_T = 3.↩ 021197546)
- type(meos_mix_reducing), parameter **meos_red_167** = meos_mix_reducing(ident1 = "C2", ident2 = "HE", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_168** = meos_mix_reducing(ident1 = "R143A", ident2 = "R134A", bibref = "E.W. Lemmon and R.T Jacobsen, J. Phys. Chem. Ref. Data, 33(2):593-620, 2004.", beta_v = 1., gamma_v = 1.0016349, beta_T = 1., gamma_T = 1.0040047)
- type(meos_mix_reducing), parameter **meos_red_169** = meos_mix_reducing(ident1 = "CO", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010) (estimated from nitrogen and oxygen)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.499)
- type(meos_mix_reducing), parameter **meos_red_170** = meos_mix_reducing(ident1 = "HE", ident2 = "NE", bibref = "E.W. Lemmon, NIST (2016)", beta_v = 1.0097, gamma_v = 0.899, beta_T = 1.168, gamma_T = 1.371)
- type(meos_mix_reducing), parameter **meos_red_171** = meos_mix_reducing(ident1 = "NC8", ident2 = "NC9", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.001357085, beta_T = 1., gamma_T = 1.↩ 000235044)
- type(meos_mix_reducing), parameter **meos_red_172** = meos_mix_reducing(ident1 = "IC4", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.01)
- type(meos_mix_reducing), parameter **meos_red_173** = meos_mix_reducing(ident1 = "CYCLOHEX", ident2 = "NC11", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010) (estimated from octane/cyclohexane and dodecane/cyclohexan", beta_v = 1., gamma_v = 1.014, beta_T = 1., gamma_T = 1.04)
- type(meos_mix_reducing), parameter **meos_red_174** = meos_mix_reducing(ident1 = "N2", ident2 = "PXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 4.99% from 265 bubble-poi", beta_v = 1., gamma_v = 1., beta_T = 0.99471, gamma_T = 1.63462)
- type(meos_mix_reducing), parameter **meos_red_175** = meos_mix_reducing(ident1 = "HE", ident2 = "AR", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_176** = meos_mix_reducing(ident1 = "CO", ident2 = "H2O", bibref = "Gernert (2013)", beta_v = 0.9404260, gamma_v = 0.7667560, beta_T = 0.956090, gamma_T = 0.8239840)
- type(meos_mix_reducing), parameter **meos_red_177** = meos_mix_reducing(ident1 = "CO2", ident2 = "ACETONE", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.1177, beta_T = 0.978377849525, gamma_T = 1.0433)
- type(meos_mix_reducing), parameter **meos_red_178** = meos_mix_reducing(ident1 = "NC9", ident2 = "CO", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.252151449, beta_T = 1., gamma_T = 1.↩ 294070556)
- type(meos_mix_reducing), parameter **meos_red_179** = meos_mix_reducing(ident1 = "O2", ident2 = "HE", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_180** = meos_mix_reducing(ident1 = "CO2", ident2 = "AR", bibref = "Neumann (2017) - Published in Lovseth et al. (2018) / see also Herrig (2018) PhD thesis", beta_v = 1.0037659, gamma_v = 1.0138330, beta_T = 0.998705, gamma_T = 1.0396748)
- type(meos_mix_reducing), parameter **meos_red_181** = meos_mix_reducing(ident1 = "NC8", ident2 = "EBZN", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.52% from 76 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.9992206, gamma_T = 0.98584)
- type(meos_mix_reducing), parameter **meos_red_182** = meos_mix_reducing(ident1 = "R23", ident2 = "F6S", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.51% from 9 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 0.994449974691, gamma_T = 0.91843)
- type(meos_mix_reducing), parameter **meos_red_183** = meos_mix_reducing(ident1 = "PRLN", ident2 = "R12", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0020049, beta_T = 1., gamma_T = 0.9887079)
- type(meos_mix_reducing), parameter **meos_red_184** = meos_mix_reducing(ident1 = "N2", ident2 = "NC10", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1., beta_T = 0.957934447, gamma_T = 1.↩ 822157123)

- type(meos_mix_reducing), parameter **meos_red_185** = meos_mix_reducing(ident1 = "N2", ident2 = "CO2", bibref = "Kunz and Wagner (2006) - original GERG-2004 mixture model used in EOS-CG!", beta_v = 0.↩ 977794634, gamma_v = 1.047578256, beta_T = 1.005894529, gamma_T = 1.107654104)
- type(meos_mix_reducing), parameter **meos_red_186** = meos_mix_reducing(ident1 = "CYCLOHEX", ident2 = "MXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.8% from 53 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.00315, gamma_T = 0.99328)
- type(meos_mix_reducing), parameter **meos_red_187** = meos_mix_reducing(ident1 = "BENZENE", ident2 = "NC12", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 2.5% from 61 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.007170043540, gamma_T = 1.02547)
- type(meos_mix_reducing), parameter **meos_red_188** = meos_mix_reducing(ident1 = "R116", ident2 = "C3", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.7% from 61 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 0.983661384405, gamma_T = 0.89173)
- type(meos_mix_reducing), parameter **meos_red_189** = meos_mix_reducing(ident1 = "C2", ident2 = "NC5", bibref = "Kunz and Wagner (2007)", beta_v = 0.993851009, gamma_v = 1.026085655, beta_T = 0.↩ 998688946, gamma_T = 1.066665676)
- type(meos_mix_reducing), parameter **meos_red_190** = meos_mix_reducing(ident1 = "NH3", ident2 = "MEOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.987, gamma_T = 1.117)
- type(meos_mix_reducing), parameter **meos_red_191** = meos_mix_reducing(ident1 = "C1", ident2 = "NC7", bibref = "Kunz and Wagner (2007)", beta_v = 0.962050831, gamma_v = 1.156655935, beta_T = 0.↩ 977431529, gamma_T = 1.379850328)
- type(meos_mix_reducing), parameter **meos_red_192** = meos_mix_reducing(ident1 = "BENZENE", ident2 = "PXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.8% from 79 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.00112, gamma_T = 1.00639)
- type(meos_mix_reducing), parameter **meos_red_193** = meos_mix_reducing(ident1 = "NC7", ident2 = "EBZN", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.52% from 151 bubble-poi", beta_v = 1., gamma_v = 1., beta_T = 0.9996002, gamma_T = 0.98665)
- type(meos_mix_reducing), parameter **meos_red_194** = meos_mix_reducing(ident1 = "R32", ident2 = "IC5", bibref = "E.W. Lemmon, NIST (2006)", beta_v = 1., gamma_v = 1.0683958, beta_T = 1., gamma_T = 0.↩ 8885477)
- type(meos_mix_reducing), parameter **meos_red_195** = meos_mix_reducing(ident1 = "C1", ident2 = "AR", bibref = "Kunz and Wagner (2007)", beta_v = 1.034630259, gamma_v = 1.014678542, beta_T = 0.↩ 990954281, gamma_T = 0.989843388)
- type(meos_mix_reducing), parameter **meos_red_196** = meos_mix_reducing(ident1 = "H2O", ident2 = "AR", bibref = "Gernert (2013)", beta_v = 0.9403980, gamma_v = 1.0509520, beta_T = 0.679104, gamma_T = 0.9210000)
- type(meos_mix_reducing), parameter **meos_red_197** = meos_mix_reducing(ident1 = "C1", ident2 = "BENZENE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.982, gamma_T = 1.266)
- type(meos_mix_reducing), parameter **meos_red_198** = meos_mix_reducing(ident1 = "R23", ident2 = "R134A", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0141263, beta_T = 1., gamma_T = 1.0105431)
- type(meos_mix_reducing), parameter **meos_red_199** = meos_mix_reducing(ident1 = "SO2", ident2 = "MEG", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.148)
- type(meos_mix_reducing), parameter **meos_red_200** = meos_mix_reducing(ident1 = "PRLN", ident2 = "IC5", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010) (estimated from trend found in C1-C10)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.035)
- type(meos_mix_reducing), parameter **meos_red_201** = meos_mix_reducing(ident1 = "C2", ident2 = "BENZENE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.081)
- type(meos_mix_reducing), parameter **meos_red_202** = meos_mix_reducing(ident1 = "BENZENE", ident2 = "NC7", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.007, beta_T = 1., gamma_T = 0.987)
- type(meos_mix_reducing), parameter **meos_red_203** = meos_mix_reducing(ident1 = "R143A", ident2 = "C3", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0000536, beta_T = 1., gamma_T = 0.9130538)

- type(meos_mix_reducing), parameter **meos_red_204** = meos_mix_reducing(ident1 = "AR", ident2 = "KR", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0037297, beta_T = 1., gamma_T = 1.↩ 0025306)
- type(meos_mix_reducing), parameter **meos_red_205** = meos_mix_reducing(ident1 = "MEOH", ident2 = "NC11", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 1.17% from 9 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.105729892302, gamma_T = 0.95529)
- type(meos_mix_reducing), parameter **meos_red_206** = meos_mix_reducing(ident1 = "CO", ident2 = "NH3", bibref = "Neumann et al. (2020)", beta_v = 0.739937224, gamma_v = 1.707, beta_T = 1.057511717, gamma_T = 0.952705)
- type(meos_mix_reducing), parameter **meos_red_207** = meos_mix_reducing(ident1 = "R41", ident2 = "TOLU", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 1.19% from 9 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 0.975390031650, gamma_T = 1.09707)
- type(meos_mix_reducing), parameter **meos_red_208** = meos_mix_reducing(ident1 = "C3", ident2 = "H2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.07400611, beta_T = 1., gamma_T = 2.↩ 308215191)
- type(meos_mix_reducing), parameter **meos_red_209** = meos_mix_reducing(ident1 = "N2", ident2 = "CY-CLOHEX", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.0177, beta_T = 1., gamma_T = 1.442)
- type(meos_mix_reducing), parameter **meos_red_210** = meos_mix_reducing(ident1 = "NC8", ident2 = "CO", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.219206702, beta_T = 1., gamma_T = 1.↩ 276565536)
- type(meos_mix_reducing), parameter **meos_red_211** = meos_mix_reducing(ident1 = "C3", ident2 = "MXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.88% from 25 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.9965519, gamma_T = 1.08363)
- type(meos_mix_reducing), parameter **meos_red_212** = meos_mix_reducing(ident1 = "NC10", ident2 = "HE", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_213** = meos_mix_reducing(ident1 = "O2", ident2 = "PRLN", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010) (estimated from nitrogen and argon)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.155)
- type(meos_mix_reducing), parameter **meos_red_214** = meos_mix_reducing(ident1 = "MEOH", ident2 = "MXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 4.↩ 08% from 11 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.99745, gamma_T = 0.88973)
- type(meos_mix_reducing), parameter **meos_red_215** = meos_mix_reducing(ident1 = "R32", ident2 = "NC4", bibref = "E.W. Lemmon, NIST (2006)", beta_v = 1., gamma_v = 0.7952703, beta_T = 1., gamma_T = 0.↩ 8997516)
- type(meos_mix_reducing), parameter **meos_red_216** = meos_mix_reducing(ident1 = "CO2", ident2 = "NC5", bibref = "Kunz and Wagner (2007)", beta_v = 1.024311498, gamma_v = 1.068406078, beta_T = 1.↩ 027000795, gamma_T = 0.979217302)
- type(meos_mix_reducing), parameter **meos_red_217** = meos_mix_reducing(ident1 = "H2S", ident2 = "MXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 3.33% from 30 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.972370008958, gamma_T = 1.10582)
- type(meos_mix_reducing), parameter **meos_red_218** = meos_mix_reducing(ident1 = "R23", ident2 = "IC4", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0362036, beta_T = 1., gamma_T = 0.↩ 9040920)
- type(meos_mix_reducing), parameter **meos_red_219** = meos_mix_reducing(ident1 = "PRLN", ident2 = "PXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 2.66% from 16 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.99082, gamma_T = 1.09426)
- type(meos_mix_reducing), parameter **meos_red_220** = meos_mix_reducing(ident1 = "ETOH", ident2 = "NC7", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 4.97% from 597 bubble-poi", beta_v = 1., gamma_v = 1., beta_T = 0.980959959569, gamma_T = 0.8981)
- type(meos_mix_reducing), parameter **meos_red_221** = meos_mix_reducing(ident1 = "ACETONE", ident2 = "TOLU", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 0.9962, beta_T = 1., gamma_T = 0.98)
- type(meos_mix_reducing), parameter **meos_red_222** = meos_mix_reducing(ident1 = "R134A", ident2 = "R142B", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0011016, beta_T = 1., gamma_T = 0.9869325)

- type(meos_mix_reducing), parameter **meos_red_223** = meos_mix_reducing(ident1 = "C2", ident2 = "NC12", bibref = "J. Watanasiri and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.24, beta_T = 0.97, gamma_T = 1.3)
- type(meos_mix_reducing), parameter **meos_red_224** = meos_mix_reducing(ident1 = "EBZN", ident2 = "OXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.↩ 57% from 48 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.001530037438, gamma_T = 0.99924)
- type(meos_mix_reducing), parameter **meos_red_225** = meos_mix_reducing(ident1 = "NC5", ident2 = "H2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.188334783, beta_T = 1., gamma_T = 2.↩ 013859174)
- type(meos_mix_reducing), parameter **meos_red_226** = meos_mix_reducing(ident1 = "IC5", ident2 = "NC9", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.028994325, beta_T = 1., gamma_T = 1.↩ 008191499)
- type(meos_mix_reducing), parameter **meos_red_227** = meos_mix_reducing(ident1 = "PXYL", ident2 = "NC10", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.89% from 57 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.001011021131, gamma_T = 0.99262)
- type(meos_mix_reducing), parameter **meos_red_228** = meos_mix_reducing(ident1 = "TOLU", ident2 = "MXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 1.↩ 35% from 70 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.0035123, gamma_T = 0.99601)
- type(meos_mix_reducing), parameter **meos_red_229** = meos_mix_reducing(ident1 = "NC7", ident2 = "NC10", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.002972346, beta_T = 1., gamma_T = 1.002229938)
- type(meos_mix_reducing), parameter **meos_red_230** = meos_mix_reducing(ident1 = "NC5", ident2 = "O2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_231** = meos_mix_reducing(ident1 = "ETOH", ident2 = "EBZN", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 1.↩ 38% from 32 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.9898638, gamma_T = 0.90463)
- type(meos_mix_reducing), parameter **meos_red_232** = meos_mix_reducing(ident1 = "C2", ident2 = "ACE-TONE", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.↩ 004722194313, gamma_T = 0.9769)
- type(meos_mix_reducing), parameter **meos_red_233** = meos_mix_reducing(ident1 = "D2", ident2 = "N2", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.43% from 7 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.015610027687, gamma_T = 1.22898)
- type(meos_mix_reducing), parameter **meos_red_234** = meos_mix_reducing(ident1 = "R116", ident2 = "PRLN", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.↩ 25% from 13 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.986183568209, gamma_T = 0.87937)
- type(meos_mix_reducing), parameter **meos_red_235** = meos_mix_reducing(ident1 = "AR", ident2 = "PRLN", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.158)
- type(meos_mix_reducing), parameter **meos_red_236** = meos_mix_reducing(ident1 = "C3", ident2 = "H2↩ S", bibref = "Kunz and Wagner (2012)", beta_v = 0.936811219, gamma_v = 1.010593999, beta_T = 0.↩ 992573556, gamma_T = 0.905829247)
- type(meos_mix_reducing), parameter **meos_red_237** = meos_mix_reducing(ident1 = "R12", ident2 = "R134A", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0003354, beta_T = 1., gamma_T = 0.9404374)
- type(meos_mix_reducing), parameter **meos_red_238** = meos_mix_reducing(ident1 = "NC7", ident2 = "CO", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.190354273, beta_T = 1., gamma_T = 1.↩ 256123503)
- type(meos_mix_reducing), parameter **meos_red_239** = meos_mix_reducing(ident1 = "R32", ident2 = "R12", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0256397, beta_T = 1., gamma_T = 0.↩ 9032201)
- type(meos_mix_reducing), parameter **meos_red_240** = meos_mix_reducing(ident1 = "NC4", ident2 = "NC5", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.01815965, beta_T = 1., gamma_T = 1.↩ 00214364)
- type(meos_mix_reducing), parameter **meos_red_241** = meos_mix_reducing(ident1 = "R143A", ident2 = "IC4", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0064788, beta_T = 1., gamma↩ _T = 0.9254452)

- type(meos_mix_reducing), parameter **meos_red_242** = meos_mix_reducing(ident1 = "MXYL", ident2 = "OXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.↩ 25% from 78 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.00073, gamma_T = 0.99997)
- type(meos_mix_reducing), parameter **meos_red_243** = meos_mix_reducing(ident1 = "C1", ident2 = "MEOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 0.713, gamma_v = 1.↩ 137, beta_T = 0.972, gamma_T = 1.308)
- type(meos_mix_reducing), parameter **meos_red_244** = meos_mix_reducing(ident1 = "H2O", ident2 = "NC12", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 0.5514, beta_T = 1.080146899978, gamma_T = 0.7908)
- type(meos_mix_reducing), parameter **meos_red_245** = meos_mix_reducing(ident1 = "N2", ident2 = "O2", bibref = "Kunz and Wagner (2006) - original GERG-2004 mixture model used in EOS-CG!", beta_v = 0.↩ 999521770, gamma_v = 0.997082328, beta_T = 0.997190589, gamma_T = 0.995157044)
- type(meos_mix_reducing), parameter **meos_red_246** = meos_mix_reducing(ident1 = "NC6", ident2 = "ETOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1.022, gamma_v = 1.026, beta_T = 1.013, gamma_T = 0.901)
- type(meos_mix_reducing), parameter **meos_red_247** = meos_mix_reducing(ident1 = "C3", ident2 = "NC9", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.199769134, beta_T = 1., gamma_T = 1.↩ 109973833)
- type(meos_mix_reducing), parameter **meos_red_248** = meos_mix_reducing(ident1 = "N2", ident2 = "IC4", bibref = "Kunz and Wagner (2007)", beta_v = 0.98641583, gamma_v = 1.100576129, beta_T = 0.99286813, gamma_T = 1.284462634)
- type(meos_mix_reducing), parameter **meos_red_249** = meos_mix_reducing(ident1 = "PRLN", ident2 = "MEOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.086, beta_T = 1., gamma_T = 0.979)
- type(meos_mix_reducing), parameter **meos_red_250** = meos_mix_reducing(ident1 = "R124", ident2 = "R142B", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0005123, beta_T = 1., gamma_T = 1.0025158)
- type(meos_mix_reducing), parameter **meos_red_251** = meos_mix_reducing(ident1 = "H2O", ident2 = "HE", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_252** = meos_mix_reducing(ident1 = "NC4", ident2 = "H2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.232939523, beta_T = 1., gamma_T = 2.↩ 509259945)
- type(meos_mix_reducing), parameter **meos_red_253** = meos_mix_reducing(ident1 = "N2", ident2 = "H2↩ S", bibref = "Kunz and Wagner (2012)", beta_v = 0.910394249, gamma_v = 1.256844157, beta_T = 1.↩ 004692366, gamma_T = 0.9601742)
- type(meos_mix_reducing), parameter **meos_red_254** = meos_mix_reducing(ident1 = "C2", ident2 = "IC5", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.045439935, beta_T = 1., gamma_T = 1.↩ 021150247)
- type(meos_mix_reducing), parameter **meos_red_255** = meos_mix_reducing(ident1 = "C1", ident2 = "H2O", bibref = "Herrig (2018) / see Herrig (2018) PhD thesis", beta_v = 1.176, gamma_v = 1.038, beta_T = 1.263, gamma_T = 0.748)
- type(meos_mix_reducing), parameter **meos_red_256** = meos_mix_reducing(ident1 = "NC5", ident2 = "NC10", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.016370338, beta_T = 1., gamma_T = 1.049035838)
- type(meos_mix_reducing), parameter **meos_red_257** = meos_mix_reducing(ident1 = "KR", ident2 = "CO2", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.↩ 948047023132, gamma_T = 1.0466)
- type(meos_mix_reducing), parameter **meos_red_258** = meos_mix_reducing(ident1 = "R12", ident2 = "CY-CLOHEX", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.03% from 2 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.038507871890, gamma_T = 0.989)
- type(meos_mix_reducing), parameter **meos_red_259** = meos_mix_reducing(ident1 = "XE", ident2 = "PRLN", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 2.04% from 51 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.009784814856, gamma_T = 1.01223)
- type(meos_mix_reducing), parameter **meos_red_260** = meos_mix_reducing(ident1 = "PRLN", ident2 = "R114", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0187337, beta_T = 1., gamma_T = 0.9744438)

- type(meos_mix_reducing), parameter **meos_red_261** = meos_mix_reducing(ident1 = "O2", ident2 = "AR", bibref = "Gernert (2013)", beta_v = 1.0065020, gamma_v = 1.0013410, beta_T = 0.999039, gamma_T = 0.9888220)
- type(meos_mix_reducing), parameter **meos_red_262** = meos_mix_reducing(ident1 = "NC4", ident2 = "ETOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.03, gamma_T = 0.95)
- type(meos_mix_reducing), parameter **meos_red_263** = meos_mix_reducing(ident1 = "C3", ident2 = "NC5", bibref = "Kunz and Wagner (2007)", beta_v = 1.044919431, gamma_v = 1.019921513, beta_T = 0.↩ 996484021, gamma_T = 1.008344412)
- type(meos_mix_reducing), parameter **meos_red_264** = meos_mix_reducing(ident1 = "R23", ident2 = "TOLU", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.65% from 9 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 0.969090002199, gamma_T = 1.07054)
- type(meos_mix_reducing), parameter **meos_red_265** = meos_mix_reducing(ident1 = "OXYL", ident2 = "NC10", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.41% from 16 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.001351824964, gamma_T = 0.99471)
- type(meos_mix_reducing), parameter **meos_red_266** = meos_mix_reducing(ident1 = "C3", ident2 = "ETOH", bibref = "E.W. Lemmon, NIST (2015)", beta_v = 1.064, gamma_v = 0.964, beta_T = 1.016, gamma_T = 0.998)
- type(meos_mix_reducing), parameter **meos_red_267** = meos_mix_reducing(ident1 = "N2", ident2 = "NC7", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.40455409, beta_T = 1., gamma_T = 1.↩ 520975334)
- type(meos_mix_reducing), parameter **meos_red_268** = meos_mix_reducing(ident1 = "IC5", ident2 = "H2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.184340443, beta_T = 1., gamma_T = 1.↩ 996386669)
- type(meos_mix_reducing), parameter **meos_red_269** = meos_mix_reducing(ident1 = "N2", ident2 = "NC8", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.186067025, beta_T = 1., gamma_T = 1.↩ 733280051)
- type(meos_mix_reducing), parameter **meos_red_270** = meos_mix_reducing(ident1 = "CO2", ident2 = "R1234ZE", bibref = "E.W. Lemmon, NIST (2013); based on simulation data from Gabriele Raabe", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.023)
- type(meos_mix_reducing), parameter **meos_red_271** = meos_mix_reducing(ident1 = "MEOH", ident2 = "EBZN", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 3.↩ 04% from 20 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.9989112, gamma_T = 0.89432)
- type(meos_mix_reducing), parameter **meos_red_272** = meos_mix_reducing(ident1 = "C1", ident2 = "IC5", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.343685343, beta_T = 1., gamma_T = 1.↩ 188899743)
- type(meos_mix_reducing), parameter **meos_red_273** = meos_mix_reducing(ident1 = "NC10", ident2 = "CO", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 0.87018496, beta_T = 1.049594632, gamma_T = 1.803567587)
- type(meos_mix_reducing), parameter **meos_red_274** = meos_mix_reducing(ident1 = "BENZENE", ident2 = "EBZN", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 3.88% from 54 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.99807, gamma_T = 1.01425)
- type(meos_mix_reducing), parameter **meos_red_275** = meos_mix_reducing(ident1 = "PRLN", ident2 = "R142B", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0035574, beta_T = 1., gamma_T = 0.9913727)
- type(meos_mix_reducing), parameter **meos_red_276** = meos_mix_reducing(ident1 = "CO2", ident2 = "R142B", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0626246, beta_T = 1., gamma_T = 0.9888000)
- type(meos_mix_reducing), parameter **meos_red_277** = meos_mix_reducing(ident1 = "NC5", ident2 = "IC5", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.000024335, beta_T = 1., gamma_T = 1.↩ 000050537)
- type(meos_mix_reducing), parameter **meos_red_278** = meos_mix_reducing(ident1 = "C1", ident2 = "PRLN", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.998, gamma_T = 1.117)
- type(meos_mix_reducing), parameter **meos_red_279** = meos_mix_reducing(ident1 = "PRLN", ident2 = "CYCLOHEX", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.57% from 12 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.999011, gamma_T = 1.04089)

- type(meos_mix_reducing), parameter **meos_red_280** = meos_mix_reducing(ident1 = "C1", ident2 = "C3", bibref = "Kunz and Wagner (2007)", beta_v = 1.00482707, gamma_v = 1.038470657, beta_T = 0.989680305, gamma_T = 1.098655531)
- type(meos_mix_reducing), parameter **meos_red_281** = meos_mix_reducing(ident1 = "SO2", ident2 = "N2", bibref = "Neumann and Herrig (2017) / see Herrig (2018) PhD thesis", beta_v = 0.903624500, gamma_v = 1.215580800, beta_T = 1.045874000, gamma_T = 1.194658800)
- type(meos_mix_reducing), parameter **meos_red_282** = meos_mix_reducing(ident1 = "PRLN", ident2 = "R115", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0082859, beta_T = 1., gamma_T = 0.9428297)
- type(meos_mix_reducing), parameter **meos_red_283** = meos_mix_reducing(ident1 = "R23", ident2 = "PRLN", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0085527, beta_T = 1., gamma_T = 0.9107001)
- type(meos_mix_reducing), parameter **meos_red_284** = meos_mix_reducing(ident1 = "NC7", ident2 = "H2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.159131722, beta_T = 1., gamma_T = 3.↩ 169143057)
- type(meos_mix_reducing), parameter **meos_red_285** = meos_mix_reducing(ident1 = "R32", ident2 = "ETOH", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 3.97% from 22 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.025031263454, gamma_T = 1.01657)
- type(meos_mix_reducing), parameter **meos_red_286** = meos_mix_reducing(ident1 = "C3", ident2 = "ACETONE", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.↩ 003109639884, gamma_T = 1.0191)
- type(meos_mix_reducing), parameter **meos_red_287** = meos_mix_reducing(ident1 = "C3", ident2 = "MEOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.992, gamma_T = 0.922)
- type(meos_mix_reducing), parameter **meos_red_288** = meos_mix_reducing(ident1 = "SO2", ident2 = "O2", bibref = "Neumann and Herrig (2017) / see Herrig (2018) PhD thesis", beta_v = 1.219246300, gamma_v = 1.660631700, beta_T = 0.927961000, gamma_T = 1.035878200)
- type(meos_mix_reducing), parameter **meos_red_289** = meos_mix_reducing(ident1 = "NC6", ident2 = "O2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_290** = meos_mix_reducing(ident1 = "NC6", ident2 = "AR", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_291** = meos_mix_reducing(ident1 = "C2", ident2 = "ETOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.179, beta_T = 0.994, gamma_T = 1.086)
- type(meos_mix_reducing), parameter **meos_red_292** = meos_mix_reducing(ident1 = "PRLN", ident2 = "C3", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.↩ 001001001001, gamma_T = 0.995)
- type(meos_mix_reducing), parameter **meos_red_293** = meos_mix_reducing(ident1 = "O2", ident2 = "MEOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.436)
- type(meos_mix_reducing), parameter **meos_red_294** = meos_mix_reducing(ident1 = "R116", ident2 = "R134A", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0010759, beta_T = 1., gamma_T = 0.9428520)
- type(meos_mix_reducing), parameter **meos_red_295** = meos_mix_reducing(ident1 = "C1", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 0.961, gamma_v = 1.085, beta_T = 0.967, gamma_T = 1.34)
- type(meos_mix_reducing), parameter **meos_red_296** = meos_mix_reducing(ident1 = "CO2", ident2 = "NC8", bibref = "Kunz and Wagner (2007)", beta_v = 1.026169373, gamma_v = 1.104043935, beta_T = 1.02969078, gamma_T = 1.074455386)
- type(meos_mix_reducing), parameter **meos_red_297** = meos_mix_reducing(ident1 = "AR", ident2 = "MEOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.436)
- type(meos_mix_reducing), parameter **meos_red_298** = meos_mix_reducing(ident1 = "CO2", ident2 = "NC7", bibref = "Kunz and Wagner (2007)", beta_v = 1.205469976, gamma_v = 1.164585914, beta_T = 1.↩ 011806317, gamma_T = 1.046169823)
- type(meos_mix_reducing), parameter **meos_red_299** = meos_mix_reducing(ident1 = "N2", ident2 = "AR", bibref = "Gernert (2013)", beta_v = 1.0066970, gamma_v = 1.0015490, beta_T = 0.999442, gamma_T = 0.9893110)

- type(meos_mix_reducing), parameter **meos_red_300** = meos_mix_reducing(ident1 = "NE", ident2 = "AR", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.051, beta_T = 1.004, gamma_T = 1.0994)
- type(meos_mix_reducing), parameter **meos_red_301** = meos_mix_reducing(ident1 = "CO", ident2 = "NC12", bibref = "J. Watanasiri and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 0.87, beta_T = 0.933, gamma_T = 1.941)
- type(meos_mix_reducing), parameter **meos_red_302** = meos_mix_reducing(ident1 = "CO2", ident2 = "R124", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.56% from 19 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.99517, gamma_T = 1.01028)
- type(meos_mix_reducing), parameter **meos_red_303** = meos_mix_reducing(ident1 = "NC8", ident2 = "PXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.2% from 51 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 0.99788, gamma_T = 0.98651)
- type(meos_mix_reducing), parameter **meos_red_304** = meos_mix_reducing(ident1 = "C2", ident2 = "H2", bibref = "Kunz and Wagner (2007)", beta_v = 0.925367171, gamma_v = 1.10607204, beta_T = 0.932969831, gamma_T = 1.902008495)
- type(meos_mix_reducing), parameter **meos_red_305** = meos_mix_reducing(ident1 = "H2S", ident2 = "BENZENE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.994, gamma_T = 1.06)
- type(meos_mix_reducing), parameter **meos_red_306** = meos_mix_reducing(ident1 = "C1", ident2 = "NC5", bibref = "Kunz and Wagner (2007)", beta_v = 0.94833012, gamma_v = 1.124508039, beta_T = 0.992127525, gamma_T = 1.249173968)
- type(meos_mix_reducing), parameter **meos_red_307** = meos_mix_reducing(ident1 = "IC4", ident2 = "NC8", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.032807063, beta_T = 1., gamma_T = 1.↩ 013945424)
- type(meos_mix_reducing), parameter **meos_red_308** = meos_mix_reducing(ident1 = "IC4", ident2 = "CYCLOHEX", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.0167)
- type(meos_mix_reducing), parameter **meos_red_309** = meos_mix_reducing(ident1 = "C1", ident2 = "NH3", bibref = "Neumann et al. (2020)", beta_v = 1.006058, gamma_v = 1.069834, beta_T = 1.022371, gamma_T = 0.940156)
- type(meos_mix_reducing), parameter **meos_red_310** = meos_mix_reducing(ident1 = "IC4", ident2 = "NC9", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.047298475, beta_T = 1., gamma_T = 1.↩ 017817492)
- type(meos_mix_reducing), parameter **meos_red_311** = meos_mix_reducing(ident1 = "CYCLOHEX", ident2 = "TOLU", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 0.9979, beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_312** = meos_mix_reducing(ident1 = "TOLU", ident2 = "EBZN", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.↩ 97% from 78 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.9971283, gamma_T = 1.00252)
- type(meos_mix_reducing), parameter **meos_red_313** = meos_mix_reducing(ident1 = "R21", ident2 = "ETOH", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.41% from 4 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.020543541490, gamma_T = 0.90692)
- type(meos_mix_reducing), parameter **meos_red_314** = meos_mix_reducing(ident1 = "R143A", ident2 = "R1234YF", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD↩ : 0.2% from 35 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 0.9997601, gamma_T = 0.99523)
- type(meos_mix_reducing), parameter **meos_red_315** = meos_mix_reducing(ident1 = "N2", ident2 = "CO", bibref = "Gernert (2013)", beta_v = 1.0000000, gamma_v = 1.0013170, beta_T = 1.002409, gamma_T = 0.9941000)
- type(meos_mix_reducing), parameter **meos_red_316** = meos_mix_reducing(ident1 = "C2", ident2 = "AR", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_317** = meos_mix_reducing(ident1 = "IC5", ident2 = "HE", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_318** = meos_mix_reducing(ident1 = "R134A", ident2 = "NC4", bibref = "E.W. Lemmon, NIST (2006)", beta_v = 1., gamma_v = 1.0047311, beta_T = 1., gamma_T = 0.8929711)
- type(meos_mix_reducing), parameter **meos_red_319** = meos_mix_reducing(ident1 = "IC5", ident2 = "H2O", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)

- type(meos_mix_reducing), parameter **meos_red_320** = meos_mix_reducing(ident1 = "CO", ident2 = "BENZENE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.435)
- type(meos_mix_reducing), parameter **meos_red_321** = meos_mix_reducing(ident1 = "NE", ident2 = "KR", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.02385, gamma_T = 1.1957)
- type(meos_mix_reducing), parameter **meos_red_322** = meos_mix_reducing(ident1 = "R134A", ident2 = "IC5", bibref = "E.W. Lemmon, NIST (2006)", beta_v = 1., gamma_v = 1.0146590, beta_T = 1., gamma_T = 0.8993586)
- type(meos_mix_reducing), parameter **meos_red_323** = meos_mix_reducing(ident1 = "NC4", ident2 = "AR", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.214638734, beta_T = 1., gamma_T = 1.245039498)
- type(meos_mix_reducing), parameter **meos_red_324** = meos_mix_reducing(ident1 = "C1", ident2 = "NC10", bibref = "Kunz and Wagner (2012)", beta_v = 1.033086292, gamma_v = 1.146089637, beta_T = 0.937777823, gamma_T = 1.568231489)
- type(meos_mix_reducing), parameter **meos_red_325** = meos_mix_reducing(ident1 = "R114", ident2 = "NC7", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.24% from 12 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.007912110064, gamma_T = 0.99663)
- type(meos_mix_reducing), parameter **meos_red_326** = meos_mix_reducing(ident1 = "H2S", ident2 = "NC12", bibref = "J. Watanasiri and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.214)
- type(meos_mix_reducing), parameter **meos_red_327** = meos_mix_reducing(ident1 = "PRLN", ident2 = "NC9", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.970873786408, gamma_T = 1.117)
- type(meos_mix_reducing), parameter **meos_red_328** = meos_mix_reducing(ident1 = "O2", ident2 = "ETOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010) (estimated from argon)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.492)
- type(meos_mix_reducing), parameter **meos_red_329** = meos_mix_reducing(ident1 = "PRLN", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.024, beta_T = 1.021450459653, gamma_T = 1.039)
- type(meos_mix_reducing), parameter **meos_red_330** = meos_mix_reducing(ident1 = "C2", ident2 = "CYCLOHEX", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.9958, gamma_T = 1.1004)
- type(meos_mix_reducing), parameter **meos_red_331** = meos_mix_reducing(ident1 = "IC5", ident2 = "AR", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_332** = meos_mix_reducing(ident1 = "C1", ident2 = "HE", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 0.881405683, beta_T = 1., gamma_T = 3.159776855)
- type(meos_mix_reducing), parameter **meos_red_333** = meos_mix_reducing(ident1 = "ACETONE", ident2 = "PXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 2.27% from 26 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.0007, gamma_T = 0.9752)
- type(meos_mix_reducing), parameter **meos_red_334** = meos_mix_reducing(ident1 = "BENZENE", ident2 = "OXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 1.58% from 30 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.01414, gamma_T = 1.00329)
- type(meos_mix_reducing), parameter **meos_red_335** = meos_mix_reducing(ident1 = "OXYL", ident2 = "NC11", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.14% from 13 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.001903616872, gamma_T = 0.99825)
- type(meos_mix_reducing), parameter **meos_red_336** = meos_mix_reducing(ident1 = "C3", ident2 = "HE", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_337** = meos_mix_reducing(ident1 = "TOLU", ident2 = "NC10", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.026)
- type(meos_mix_reducing), parameter **meos_red_338** = meos_mix_reducing(ident1 = "O2", ident2 = "KR", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0043319, beta_T = 1., gamma_T = 0.9729000)
- type(meos_mix_reducing), parameter **meos_red_339** = meos_mix_reducing(ident1 = "CYCLOHEX", ident2 = "NC7", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.0015, beta_T = 1., gamma_T = 1.)

- type(meos_mix_reducing), parameter **meos_red_340** = meos_mix_reducing(ident1 = "IC5", ident2 = "MEOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.2935310, beta_T = 1., gamma_T = 0.8541581)
- type(meos_mix_reducing), parameter **meos_red_341** = meos_mix_reducing(ident1 = "NC5", ident2 = "ACETONE", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.005833836250, gamma_T = 0.9221)
- type(meos_mix_reducing), parameter **meos_red_342** = meos_mix_reducing(ident1 = "CO2", ident2 = "R32", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0058521, beta_T = 1., gamma_T = 0.9978225)
- type(meos_mix_reducing), parameter **meos_red_343** = meos_mix_reducing(ident1 = "IC4", ident2 = "H2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.147595688, beta_T = 1., gamma_T = 1.895305393)
- type(meos_mix_reducing), parameter **meos_red_344** = meos_mix_reducing(ident1 = "IC5", ident2 = "NC10", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.039372957, beta_T = 1., gamma_T = 1.010825138)
- type(meos_mix_reducing), parameter **meos_red_345** = meos_mix_reducing(ident1 = "MEOH", ident2 = "CYCLOHEX", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.0530481, beta_T = 1., gamma_T = 0.8680742)
- type(meos_mix_reducing), parameter **meos_red_346** = meos_mix_reducing(ident1 = "O2", ident2 = "H2O", bibref = "Gernert (2013)", beta_v = 1.0281970, gamma_v = 0.8734600, beta_T = 1.253060, gamma_T = 0.8078420)
- type(meos_mix_reducing), parameter **meos_red_347** = meos_mix_reducing(ident1 = "C2", ident2 = "NC9", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.14353473, beta_T = 1., gamma_T = 1.005603303)
- type(meos_mix_reducing), parameter **meos_red_348** = meos_mix_reducing(ident1 = "NC4", ident2 = "NC10", bibref = "Kunz and Wagner (2012)", beta_v = 0.976951968, gamma_v = 1.027845529, beta_T = 0.993688386, gamma_T = 1.076466918)
- type(meos_mix_reducing), parameter **meos_red_349** = meos_mix_reducing(ident1 = "MXYL", ident2 = "NC9", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.26% from 12 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.002270041417, gamma_T = 0.98031)
- type(meos_mix_reducing), parameter **meos_red_350** = meos_mix_reducing(ident1 = "R32", ident2 = "TOLU", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 1.49% from 9 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.015980045339, gamma_T = 0.98173)
- type(meos_mix_reducing), parameter **meos_red_351** = meos_mix_reducing(ident1 = "R32", ident2 = "R115", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0424065, beta_T = 1., gamma_T = 0.8807734)
- type(meos_mix_reducing), parameter **meos_red_352** = meos_mix_reducing(ident1 = "C3", ident2 = "CYCLOHEX", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.0023, gamma_T = 1.035)
- type(meos_mix_reducing), parameter **meos_red_353** = meos_mix_reducing(ident1 = "NC5", ident2 = "H2O", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 0.95667731, beta_T = 1., gamma_T = 0.447666011)
- type(meos_mix_reducing), parameter **meos_red_354** = meos_mix_reducing(ident1 = "MEOH", ident2 = "NC6", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 0.9683251, beta_T = 1., gamma_T = 0.8780631)
- type(meos_mix_reducing), parameter **meos_red_355** = meos_mix_reducing(ident1 = "O2", ident2 = "CO", bibref = "Gernert (2013) - no adjusted parameters", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_356** = meos_mix_reducing(ident1 = "H2", ident2 = "AR", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_357** = meos_mix_reducing(ident1 = "NC4", ident2 = "HE", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_358** = meos_mix_reducing(ident1 = "SO2", ident2 = "C1", bibref = "Zosimenko and Herrig (2017) / see Herrig (2018) PhD thesis ", beta_v = 1.3155340, gamma_v = 1.1195433, beta_T = 0.999432, gamma_T = 1.1157135)
- type(meos_mix_reducing), parameter **meos_red_359** = meos_mix_reducing(ident1 = "CYCLOHEX", ident2 = "NC8", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.0069, beta_T = 1., gamma_T = 1.008)

- type(meos_mix_reducing), parameter **meos_red_360** = meos_mix_reducing(ident1 = "NC6", ident2 = "BENZENE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 0.999, beta_T = 1., gamma_T = 0.991)
- type(meos_mix_reducing), parameter **meos_red_361** = meos_mix_reducing(ident1 = "C2", ident2 = "O2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_362** = meos_mix_reducing(ident1 = "NC9", ident2 = "AR", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_363** = meos_mix_reducing(ident1 = "NC6", ident2 = "OXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 1.3% from 12 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.00096, gamma_T = 1.00154)
- type(meos_mix_reducing), parameter **meos_red_364** = meos_mix_reducing(ident1 = "NC5", ident2 = "CO", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.119954454, beta_T = 1., gamma_T = 1.↩ 206043295)
- type(meos_mix_reducing), parameter **meos_red_365** = meos_mix_reducing(ident1 = "C3", ident2 = "NC8", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.102764612, beta_T = 1., gamma_T = 1.↩ 063694129)
- type(meos_mix_reducing), parameter **meos_red_366** = meos_mix_reducing(ident1 = "NC5", ident2 = "NC9", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.034910633, beta_T = 1., gamma_T = 1.↩ 103421755)
- type(meos_mix_reducing), parameter **meos_red_367** = meos_mix_reducing(ident1 = "D2", ident2 = "NE", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 1.22% from 78 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.98882, gamma_T = 0.83393)
- type(meos_mix_reducing), parameter **meos_red_368** = meos_mix_reducing(ident1 = "R32", ident2 = "R134A", bibref = "E.W. Lemmon and R.T Jacobsen, J. Phys. Chem. Ref. Data, 33(2):593-620, 2004.", beta_v = 1., gamma_v = 1.0137207, beta_T = 1., gamma_T = 1.0114076)
- type(meos_mix_reducing), parameter **meos_red_369** = meos_mix_reducing(ident1 = "C3", ident2 = "C3_1", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 1.85% from 13 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.0024359, gamma_T = 0.98041)
- type(meos_mix_reducing), parameter **meos_red_370** = meos_mix_reducing(ident1 = "PRLN", ident2 = "NH3", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.9721, gamma_T = 0.8483)
- type(meos_mix_reducing), parameter **meos_red_371** = meos_mix_reducing(ident1 = "R23", ident2 = "R12", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0187876, beta_T = 1., gamma_T = 0.↩ 9266031)
- type(meos_mix_reducing), parameter **meos_red_372** = meos_mix_reducing(ident1 = "NC7", ident2 = "MXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.37% from 48 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.99846, gamma_T = 0.9896)
- type(meos_mix_reducing), parameter **meos_red_373** = meos_mix_reducing(ident1 = "H2S", ident2 = "ACETONE", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.01% from 2 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.042122595302, gamma_T = 1.04871)
- type(meos_mix_reducing), parameter **meos_red_374** = meos_mix_reducing(ident1 = "NC4", ident2 = "NC6", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.034995284, beta_T = 1., gamma_T = 1.↩ 00915706)
- type(meos_mix_reducing), parameter **meos_red_375** = meos_mix_reducing(ident1 = "NH3", ident2 = "ETOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.101)
- type(meos_mix_reducing), parameter **meos_red_376** = meos_mix_reducing(ident1 = "H2S", ident2 = "H2↩ O", bibref = "Herrig and Koerber (2018) / see Herrig (2018) PhD thesis ", beta_v = 0.945818, gamma_v = 1.189702, beta_T = 0.960526, gamma_T = 0.924026)
- type(meos_mix_reducing), parameter **meos_red_377** = meos_mix_reducing(ident1 = "IC4", ident2 = "O2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_378** = meos_mix_reducing(ident1 = "NC8", ident2 = "NC12", bibref = "J. Watanasiri and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.99884, gamma_T = 1.025)
- type(meos_mix_reducing), parameter **meos_red_379** = meos_mix_reducing(ident1 = "H2", ident2 = "D2", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.87% from 150 bubble-poi", beta_v = 1., gamma_v = 1., beta_T = 1.0087153, gamma_T = 0.99511)

- type(meos_mix_reducing), parameter **meos_red_380** = meos_mix_reducing(ident1 = "R12", ident2 = "R11", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0018062, beta_T = 1., gamma_T = 1.↩ 0044593)
- type(meos_mix_reducing), parameter **meos_red_381** = meos_mix_reducing(ident1 = "NH3", ident2 = "IC4", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010) (estimated from propane/ammonia and butane/ammonia)", beta_v = 1., gamma_v = 1., beta_T = 1.048657718121, gamma_T = 0.7986)
- type(meos_mix_reducing), parameter **meos_red_382** = meos_mix_reducing(ident1 = "O2", ident2 = "N2↩ O", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.9525, gamma_T = 1.1068)
- type(meos_mix_reducing), parameter **meos_red_383** = meos_mix_reducing(ident1 = "NH3", ident2 = "R134A", bibref = "I.H. Bell, NIST (2018)", beta_v = 1., gamma_v = 1., beta_T = 1.055547, gamma_T = 0.935232)
- type(meos_mix_reducing), parameter **meos_red_384** = meos_mix_reducing(ident1 = "NC10", ident2 = "AR", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_385** = meos_mix_reducing(ident1 = "AR", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010) (estimated from oxygen)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.45)
- type(meos_mix_reducing), parameter **meos_red_386** = meos_mix_reducing(ident1 = "CO2", ident2 = "OXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 3.93% from 102 bubble-poi", beta_v = 1., gamma_v = 1., beta_T = 1.00114, gamma_T = 1.10292)
- type(meos_mix_reducing), parameter **meos_red_387** = meos_mix_reducing(ident1 = "C3", ident2 = "R115", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0043338, beta_T = 1., gamma_T = 0.↩ 9432827)
- type(meos_mix_reducing), parameter **meos_red_388** = meos_mix_reducing(ident1 = "H2", ident2 = "O2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_389** = meos_mix_reducing(ident1 = "IC5", ident2 = "NC8", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.017880545, beta_T = 1., gamma_T = 1.↩ 00564748)
- type(meos_mix_reducing), parameter **meos_red_390** = meos_mix_reducing(ident1 = "H2", ident2 = "H2O", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_391** = meos_mix_reducing(ident1 = "H2S", ident2 = "AR", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_392** = meos_mix_reducing(ident1 = "R143A", ident2 = "NC4", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.87% from 112 bubble-poi", beta_v = 1., gamma_v = 1., beta_T = 0.999070864096, gamma_T = 0.91912)
- type(meos_mix_reducing), parameter **meos_red_393** = meos_mix_reducing(ident1 = "ACETONE", ident2 = "BENZENE", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 0.9826, beta_T = 1., gamma_T = 0.984)
- type(meos_mix_reducing), parameter **meos_red_394** = meos_mix_reducing(ident1 = "C2", ident2 = "H2↩ S", bibref = "Kunz and Wagner (2012)", beta_v = 1.010817909, gamma_v = 1.030988277, beta_T = 0.↩ 990197354, gamma_T = 0.90273666)
- type(meos_mix_reducing), parameter **meos_red_395** = meos_mix_reducing(ident1 = "ETOH", ident2 = "MEG", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 0.9095, beta_T = 1., gamma_T = 1.0174)
- type(meos_mix_reducing), parameter **meos_red_396** = meos_mix_reducing(ident1 = "SO2", ident2 = "H2", bibref = "Herrig (2018) - Linear Combining Rules / see Herrig (2018) PhD thesis", beta_v = 1.000000000, gamma_v = 1.035170959, beta_T = 1.000000000, gamma_T = 1.940852069)
- type(meos_mix_reducing), parameter **meos_red_397** = meos_mix_reducing(ident1 = "NC6", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 0.988)
- type(meos_mix_reducing), parameter **meos_red_398** = meos_mix_reducing(ident1 = "NC4", ident2 = "MEOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.037, gamma_T = 0.872)
- type(meos_mix_reducing), parameter **meos_red_399** = meos_mix_reducing(ident1 = "R114", ident2 = "BENZENE", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.38% from 12 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.998821390759, gamma_T = 0.95788)

- type(meos_mix_reducing), parameter **meos_red_400** = meos_mix_reducing(ident1 = "IC4", ident2 = "HE", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_401** = meos_mix_reducing(ident1 = "NC6", ident2 = "CO", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.155145836, beta_T = 1., gamma_T = 1.↩ 233272781)
- type(meos_mix_reducing), parameter **meos_red_402** = meos_mix_reducing(ident1 = "N2", ident2 = "R14", bibref = "E.W. Lemmon, NIST (2014)", beta_v = 1., gamma_v = 1.1693, beta_T = 1., gamma_T = 1.0674)
- type(meos_mix_reducing), parameter **meos_red_403** = meos_mix_reducing(ident1 = "ETOH", ident2 = "NC10", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.059, beta_T = 1.004016064257, gamma_T = 0.894)
- type(meos_mix_reducing), parameter **meos_red_404** = meos_mix_reducing(ident1 = "NC6", ident2 = "PXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.28% from 27 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.00071, gamma_T = 0.99711)
- type(meos_mix_reducing), parameter **meos_red_405** = meos_mix_reducing(ident1 = "NC6", ident2 = "NC9", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.02076168, beta_T = 1., gamma_T = 1.↩ 055369591)
- type(meos_mix_reducing), parameter **meos_red_406** = meos_mix_reducing(ident1 = "NC6", ident2 = "CY-CLOHEX", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 0.9979, beta_T = 1., gamma_T = 1.002)
- type(meos_mix_reducing), parameter **meos_red_407** = meos_mix_reducing(ident1 = "C1", ident2 = "C2", bibref = "Kunz and Wagner (2007)", beta_v = 0.997547866, gamma_v = 1.006617867, beta_T = 0.↩ 996336508, gamma_T = 1.049707697)
- type(meos_mix_reducing), parameter **meos_red_408** = meos_mix_reducing(ident1 = "NH3", ident2 = "NC6", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.↩ 130454442686, gamma_T = 0.8776)
- type(meos_mix_reducing), parameter **meos_red_409** = meos_mix_reducing(ident1 = "PRLN", ident2 = "NC7", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.008064516129, gamma_T = 1.082)
- type(meos_mix_reducing), parameter **meos_red_410** = meos_mix_reducing(ident1 = "NC4", ident2 = "NC9", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.049219137, beta_T = 1., gamma_T = 1.↩ 014096448)
- type(meos_mix_reducing), parameter **meos_red_411** = meos_mix_reducing(ident1 = "R23", ident2 = "R11", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0321197, beta_T = 1., gamma_T = 0.↩ 9362767)
- type(meos_mix_reducing), parameter **meos_red_412** = meos_mix_reducing(ident1 = "C1", ident2 = "NC9", bibref = "Kunz and Wagner (2012)", beta_v = 1.002852287, gamma_v = 1.141895355, beta_T = 0.↩ 947716769, gamma_T = 1.528532478)
- type(meos_mix_reducing), parameter **meos_red_413** = meos_mix_reducing(ident1 = "PXYL", ident2 = "NC9", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.22% from 11 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.999390371873, gamma_T = 0.99387)
- type(meos_mix_reducing), parameter **meos_red_414** = meos_mix_reducing(ident1 = "C1", ident2 = "NC6", bibref = "Kunz and Wagner (2007)", beta_v = 0.958015294, gamma_v = 1.052643846, beta_T = 0.↩ 981844797, gamma_T = 1.330570181)
- type(meos_mix_reducing), parameter **meos_red_415** = meos_mix_reducing(ident1 = "NC9", ident2 = "NC10", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.00081052, beta_T = 1., gamma_T = 1.000182392)
- type(meos_mix_reducing), parameter **meos_red_416** = meos_mix_reducing(ident1 = "NC4", ident2 = "H2↩ S", bibref = "Kunz and Wagner (2012)", beta_v = 0.908113163, gamma_v = 1.033366041, beta_T = 0.↩ 985962886, gamma_T = 0.926156602)
- type(meos_mix_reducing), parameter **meos_red_417** = meos_mix_reducing(ident1 = "O2", ident2 = "H2S", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_418** = meos_mix_reducing(ident1 = "NC5", ident2 = "AR", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_419** = meos_mix_reducing(ident1 = "NC8", ident2 = "H2S", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_420** = meos_mix_reducing(ident1 = "MXYL", ident2 = "NC10", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 2.15% from 44 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.999650122457, gamma_T = 0.99369)

- type(meos_mix_reducing), parameter **meos_red_421** = meos_mix_reducing(ident1 = "CO2", ident2 = "R12", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.93% from 25 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.01811, gamma_T = 0.9576)
- type(meos_mix_reducing), parameter **meos_red_422** = meos_mix_reducing(ident1 = "R12", ident2 = "R142B", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0002214, beta_T = 1., gamma_T = 0.9777321)
- type(meos_mix_reducing), parameter **meos_red_423** = meos_mix_reducing(ident1 = "NC4", ident2 = "ACETONE", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 0.8906, beta_T = 1.016156894625, gamma_T = 0.9452)
- type(meos_mix_reducing), parameter **meos_red_424** = meos_mix_reducing(ident1 = "R32", ident2 = "C3", bibref = "E.W. Lemmon, NIST (2006)", beta_v = 1., gamma_v = 1.0198058, beta_T = 1., gamma_T = 0.↩ 8616194)
- type(meos_mix_reducing), parameter **meos_red_425** = meos_mix_reducing(ident1 = "XE", ident2 = "MEOH", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.63% from 13 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.068855683106, gamma_T = 0.77335)
- type(meos_mix_reducing), parameter **meos_red_426** = meos_mix_reducing(ident1 = "NC9", ident2 = "NC12", bibref = "J. Watanasiri and E.W. Lemmon, NIST (2010) (estimated from octane and decane mixed with dodecane)", beta_v = 1., gamma_v = 1.005, beta_T = 0.999, gamma_T = 1.01)
- type(meos_mix_reducing), parameter **meos_red_427** = meos_mix_reducing(ident1 = "C1", ident2 = "KR", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0003826, beta_T = 1., gamma_T = 0.↩ 9930853)
- type(meos_mix_reducing), parameter **meos_red_428** = meos_mix_reducing(ident1 = "NE", ident2 = "O2", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0260610, beta_T = 1., gamma_T = 1.↩ 1554168)
- type(meos_mix_reducing), parameter **meos_red_429** = meos_mix_reducing(ident1 = "PRLN", ident2 = "NC8", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.979431929481, gamma_T = 1.097)
- type(meos_mix_reducing), parameter **meos_red_430** = meos_mix_reducing(ident1 = "CO2", ident2 = "CO", bibref = "Souza and Herrig (2018) / see Souza et al. (2019) and Herrig (2018) PhD thesis ", beta_v = 1.↩ 0338017, gamma_v = 1.0001623, beta_T = 0.989782, gamma_T = 1.1621298)
- type(meos_mix_reducing), parameter **meos_red_431** = meos_mix_reducing(ident1 = "C2", ident2 = "NC11", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 2.01% from 19 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.9430937, gamma_T = 1.27938)
- type(meos_mix_reducing), parameter **meos_red_432** = meos_mix_reducing(ident1 = "IC4", ident2 = "AR", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_433** = meos_mix_reducing(ident1 = "MEOH", ident2 = "NC10", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 0.957, beta_T = 1.095290251917, gamma_T = 1.047)
- type(meos_mix_reducing), parameter **meos_red_434** = meos_mix_reducing(ident1 = "N2", ident2 = "H2O", bibref = "Gernert (2013)", beta_v = 0.9262450, gamma_v = 0.7334430, beta_T = 1.048054, gamma_T = 0.8051470)
- type(meos_mix_reducing), parameter **meos_red_435** = meos_mix_reducing(ident1 = "SO2", ident2 = "BENZENE", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.0266, beta_T = 1.015847216579, gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_436** = meos_mix_reducing(ident1 = "TOLU", ident2 = "PXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.54% from 53 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.0008507, gamma_T = 0.99999)
- type(meos_mix_reducing), parameter **meos_red_437** = meos_mix_reducing(ident1 = "C3", ident2 = "R134↩ A", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0000015, beta_T = 1., gamma_T = 0.9002394)
- type(meos_mix_reducing), parameter **meos_red_438** = meos_mix_reducing(ident1 = "NH3", ident2 = "NC7", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010) (estimated from hexane/ammonia and octane/ammonia)", beta_v = 1., gamma_v = 1., beta_T = 1.178828244725, gamma_T = 0.9726)
- type(meos_mix_reducing), parameter **meos_red_439** = meos_mix_reducing(ident1 = "CO2", ident2 = "MEOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1.026, gamma_v = 1.236, beta_T = 0.996, gamma_T = 1.015)

- type(meos_mix_reducing), parameter **meos_red_440** = meos_mix_reducing(ident1 = "EBZN", ident2 = "NC9", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.21% from 50 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.003750010038, gamma_T = 0.98564)
- type(meos_mix_reducing), parameter **meos_red_441** = meos_mix_reducing(ident1 = "R14", ident2 = "TOLU", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 1.69% from 8 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 0.920319984215, gamma_T = 1.18828)
- type(meos_mix_reducing), parameter **meos_red_442** = meos_mix_reducing(ident1 = "ACETONE", ident2 = "NC12", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.0915, beta_T = 1.044, gamma_T = 0.995)
- type(meos_mix_reducing), parameter **meos_red_443** = meos_mix_reducing(ident1 = "BENZENE", ident2 = "NC10", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.008064516129, gamma_T = 1.052)
- type(meos_mix_reducing), parameter **meos_red_444** = meos_mix_reducing(ident1 = "NC5", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 0.992)
- type(meos_mix_reducing), parameter **meos_red_445** = meos_mix_reducing(ident1 = "NH3", ident2 = "BENZENE", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.↩ 029654036244, gamma_T = 0.9299)
- type(meos_mix_reducing), parameter **meos_red_446** = meos_mix_reducing(ident1 = "IC4", ident2 = "BENZENE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 0.992)
- type(meos_mix_reducing), parameter **meos_red_447** = meos_mix_reducing(ident1 = "NH3", ident2 = "NC4", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.↩ 055631795630, gamma_T = 0.8067)
- type(meos_mix_reducing), parameter **meos_red_448** = meos_mix_reducing(ident1 = "H2S", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.075)
- type(meos_mix_reducing), parameter **meos_red_449** = meos_mix_reducing(ident1 = "C2", ident2 = "NC10", bibref = "Kunz and Wagner (2012)", beta_v = 0.995676258, gamma_v = 1.098361281, beta_T = 0.↩ 970918061, gamma_T = 1.237191558)
- type(meos_mix_reducing), parameter **meos_red_450** = meos_mix_reducing(ident1 = "ETOH", ident2 = "OXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.↩ 93% from 14 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.98697, gamma_T = 0.90681)
- type(meos_mix_reducing), parameter **meos_red_451** = meos_mix_reducing(ident1 = "PRLN", ident2 = "NC6", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.021450459653, gamma_T = 1.068)
- type(meos_mix_reducing), parameter **meos_red_452** = meos_mix_reducing(ident1 = "H2S", ident2 = "CYCLOHEX", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.011, gamma_T = 1.0138)
- type(meos_mix_reducing), parameter **meos_red_453** = meos_mix_reducing(ident1 = "NC6", ident2 = "HE", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_454** = meos_mix_reducing(ident1 = "NC9", ident2 = "H2↩ S", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.082905109, beta_T = 1., gamma_T = 1.086557826)
- type(meos_mix_reducing), parameter **meos_red_455** = meos_mix_reducing(ident1 = "CO2", ident2 = "SO2", bibref = "Neumann and Herrig (2017) / see Herrig (2018) PhD thesis", beta_v = 0.8898650, gamma_v = 1.↩ 0057783, beta_T = 1.020063, gamma_T = 1.0079753)
- type(meos_mix_reducing), parameter **meos_red_456** = meos_mix_reducing(ident1 = "N2", ident2 = "SO2", bibref = "Bell & Herrig (2015)", beta_v = 1.000000000, gamma_v = 1.000000000, beta_T = 1.004300000, gamma_T = 1.220360000)
- type(meos_mix_reducing), parameter **meos_red_457** = meos_mix_reducing(ident1 = "NH3", ident2 = "IC5", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010) (estimated from butane/ammonia and pentane/ammonia)", beta_v = 1., gamma_v = 1., beta_T = 1.106194690265, gamma_T = 0.8385)
- type(meos_mix_reducing), parameter **meos_red_458** = meos_mix_reducing(ident1 = "R1234YF", ident2 = "R1234ZE", bibref = "E.W. Lemmon, NIST (2015); fit of data from Honeywell", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 0.987)

- type(meos_mix_reducing), parameter **meos_red_459** = meos_mix_reducing(ident1 = "ETOH", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.982318271120, gamma_T = 0.904)
- type(meos_mix_reducing), parameter **meos_red_460** = meos_mix_reducing(ident1 = "C3", ident2 = "IC5", bibref = "Kunz and Wagner (2007)", beta_v = 1.040459289, gamma_v = 0.999432118, beta_T = 0.↩ 994364425, gamma_T = 1.0032695)
- type(meos_mix_reducing), parameter **meos_red_461** = meos_mix_reducing(ident1 = "CO", ident2 = "AR", bibref = "Kunz and Wagner (2006) - original GERG-2004 mixture model used in EOS-CG!", beta_v = 1.↩ 000000000, gamma_v = 1.159720623, beta_T = 1.000000000, gamma_T = 0.954215746)
- type(meos_mix_reducing), parameter **meos_red_462** = meos_mix_reducing(ident1 = "IC5", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_463** = meos_mix_reducing(ident1 = "BENZENE", ident2 = "NC8", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.016, beta_T = 1., gamma_T = 0.994)
- type(meos_mix_reducing), parameter **meos_red_464** = meos_mix_reducing(ident1 = "R134A", ident2 = "R124", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0031148, beta_T = 1., gamma_T = 0.9874736)
- type(meos_mix_reducing), parameter **meos_red_465** = meos_mix_reducing(ident1 = "NC6", ident2 = "H2↩ S", bibref = "Kunz and Wagner (2012)", beta_v = 0.754473958, gamma_v = 1.339283552, beta_T = 0.↩ 985891113, gamma_T = 0.956075596)
- type(meos_mix_reducing), parameter **meos_red_466** = meos_mix_reducing(ident1 = "BENZENE", ident2 = "CYCLOHEX", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 0.9933)
- type(meos_mix_reducing), parameter **meos_red_467** = meos_mix_reducing(ident1 = "C2", ident2 = "NC8", bibref = "Kunz and Wagner (2007)", beta_v = 1.007469726, gamma_v = 1.071917985, beta_T = 0.↩ 984068272, gamma_T = 1.168636194)
- type(meos_mix_reducing), parameter **meos_red_468** = meos_mix_reducing(ident1 = "C2", ident2 = "R23", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.47% from 44 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.00597, gamma_T = 0.87082)
- type(meos_mix_reducing), parameter **meos_red_469** = meos_mix_reducing(ident1 = "C3", ident2 = "NC7", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.079648053, beta_T = 1., gamma_T = 1.↩ 050044169)
- type(meos_mix_reducing), parameter **meos_red_470** = meos_mix_reducing(ident1 = "PRLN", ident2 = "ETOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.011, gamma_T = 1.022)
- type(meos_mix_reducing), parameter **meos_red_471** = meos_mix_reducing(ident1 = "NC6", ident2 = "NC12", bibref = "J. Watanasiri and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.038, beta_T = 0.991, gamma_T = 1.027)
- type(meos_mix_reducing), parameter **meos_red_472** = meos_mix_reducing(ident1 = "CO2", ident2 = "R134A", bibref = "E.W. Lemmon, NIST (2013)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.008)
- type(meos_mix_reducing), parameter **meos_red_473** = meos_mix_reducing(ident1 = "CO2", ident2 = "NC12", bibref = "J. Watanasiri and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.258)
- type(meos_mix_reducing), parameter **meos_red_474** = meos_mix_reducing(ident1 = "MEOH", ident2 = "MEG", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1.0723, gamma_v = 0.8637, beta_T = 0.9942, gamma_T = 1.0691)
- type(meos_mix_reducing), parameter **meos_red_475** = meos_mix_reducing(ident1 = "IC4", ident2 = "NC6", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.010493989, beta_T = 1., gamma_T = 1.↩ 006018054)
- type(meos_mix_reducing), parameter **meos_red_476** = meos_mix_reducing(ident1 = "CYCLOHEX", ident2 = "OXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.73% from 39 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.00392, gamma_T = 0.99006)
- type(meos_mix_reducing), parameter **meos_red_477** = meos_mix_reducing(ident1 = "R134A", ident2 = "IC4", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0051783, beta_T = 1., gamma↩ _T = 0.8937027)

- type(meos_mix_reducing), parameter **meos_red_478** = meos_mix_reducing(ident1 = "CO", ident2 = "CYCLOHEX", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.9917, gamma_T = 1.4608)
- type(meos_mix_reducing), parameter **meos_red_479** = meos_mix_reducing(ident1 = "SO2", ident2 = "AR", bibref = "Herrig (2018) - Linear Combining Rules / see Herrig (2018) PhD thesis", beta_v = 1.000000000, gamma_v = 1.021291140, beta_T = 1.000000000, gamma_T = 1.141020219)
- type(meos_mix_reducing), parameter **meos_red_480** = meos_mix_reducing(ident1 = "R116", ident2 = "R143A", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.↩ 38% from 78 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.01003, gamma_T = 0.93499)
- type(meos_mix_reducing), parameter **meos_red_481** = meos_mix_reducing(ident1 = "N2", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.524)
- type(meos_mix_reducing), parameter **meos_red_482** = meos_mix_reducing(ident1 = "CO2", ident2 = "HE", bibref = "Kunz and Wagner (2007)", beta_v = 0.846647561, gamma_v = 0.864141549, beta_T = 0.76837763, gamma_T = 3.207456948)
- type(meos_mix_reducing), parameter **meos_red_483** = meos_mix_reducing(ident1 = "BENZENE", ident2 = "NC9", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.005025125628, gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_484** = meos_mix_reducing(ident1 = "XE", ident2 = "C2", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0034307, beta_T = 1., gamma_T = 1.↩ 0112641)
- type(meos_mix_reducing), parameter **meos_red_485** = meos_mix_reducing(ident1 = "R134A", ident2 = "R1234ZE", bibref = "E.W. Lemmon, NIST (2015); fit of data from Honeywell", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 0.992)
- type(meos_mix_reducing), parameter **meos_red_486** = meos_mix_reducing(ident1 = "CO2", ident2 = "EBZN", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 1.99% from 95 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.00091, gamma_T = 1.09145)
- type(meos_mix_reducing), parameter **meos_red_487** = meos_mix_reducing(ident1 = "IC4", ident2 = "CO", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.087272232, beta_T = 1., gamma_T = 1.↩ 161390082)
- type(meos_mix_reducing), parameter **meos_red_488** = meos_mix_reducing(ident1 = "N2", ident2 = "NC9", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.100405929, beta_T = 0.95637945, gamma_T = 1.749119996)
- type(meos_mix_reducing), parameter **meos_red_489** = meos_mix_reducing(ident1 = "IC5", ident2 = "CYCLOHEX", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010) (estimated from pentane/cyclohexane)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 0.9961)
- type(meos_mix_reducing), parameter **meos_red_490** = meos_mix_reducing(ident1 = "C3", ident2 = "IC4", bibref = "Kunz and Wagner (2007)", beta_v = 0.999243146, gamma_v = 1.001156119, beta_T = 0.↩ 998012298, gamma_T = 1.005250774)
- type(meos_mix_reducing), parameter **meos_red_491** = meos_mix_reducing(ident1 = "C3", ident2 = "BENZENE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.018)
- type(meos_mix_reducing), parameter **meos_red_492** = meos_mix_reducing(ident1 = "IC5", ident2 = "CO", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.116694577, beta_T = 1., gamma_T = 1.↩ 199326059)
- type(meos_mix_reducing), parameter **meos_red_493** = meos_mix_reducing(ident1 = "NC7", ident2 = "HE", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_494** = meos_mix_reducing(ident1 = "C3", ident2 = "O2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_495** = meos_mix_reducing(ident1 = "CO", ident2 = "PXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 1.19% from 30 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.981810006018, gamma_T = 1.62153)
- type(meos_mix_reducing), parameter **meos_red_496** = meos_mix_reducing(ident1 = "R12", ident2 = "NC8", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.02% from 2 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.003482082827, gamma_T = 1.06084)
- type(meos_mix_reducing), parameter **meos_red_497** = meos_mix_reducing(ident1 = "EBZN", ident2 = "MXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.↩ 35% from 125 bubble-poi", beta_v = 1., gamma_v = 1., beta_T = 1.000120014402, gamma_T = 0.99939)

- type(meos_mix_reducing), parameter **meos_red_498** = meos_mix_reducing(ident1 = "R32", ident2 = "R1234ZE", bibref = "R. Akasaka, FPE, 2013, DOI:10.1016/j.fluid.2013.07.057", beta_v = 1.00586, gamma_v = 0.982707, beta_T = 1.00343, gamma_T = 0.977857)
- type(meos_mix_reducing), parameter **meos_red_499** = meos_mix_reducing(ident1 = "ACETONE", ident2 = "NC10", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.0563, beta_T = 1.044, gamma_T = 0.9779)
- type(meos_mix_reducing), parameter **meos_red_500** = meos_mix_reducing(ident1 = "C3", ident2 = "NC6", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.057872566, beta_T = 1., gamma_T = 1.025657518)
- type(meos_mix_reducing), parameter **meos_red_501** = meos_mix_reducing(ident1 = "SO2", ident2 = "TOLU", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 2.74% from 27 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.011330031476, gamma_T = 1.02114)
- type(meos_mix_reducing), parameter **meos_red_502** = meos_mix_reducing(ident1 = "N2", ident2 = "HE", bibref = "Kunz and Wagner (2007)", beta_v = 0.969501055, gamma_v = 0.932629867, beta_T = 0.692868765, gamma_T = 1.47183158)
- type(meos_mix_reducing), parameter **meos_red_503** = meos_mix_reducing(ident1 = "NC8", ident2 = "H2O", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 0.599484191, beta_T = 1., gamma_T = 0.662072469)
- type(meos_mix_reducing), parameter **meos_red_504** = meos_mix_reducing(ident1 = "PRLN", ident2 = "IC4", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.971817298348, gamma_T = 1.014)
- type(meos_mix_reducing), parameter **meos_red_505** = meos_mix_reducing(ident1 = "MEOH", ident2 = "ETOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.012, beta_T = 1., gamma_T = 0.999)
- type(meos_mix_reducing), parameter **meos_red_506** = meos_mix_reducing(ident1 = "IC4", ident2 = "NC5", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.002779804, beta_T = 1., gamma_T = 1.002495889)
- type(meos_mix_reducing), parameter **meos_red_507** = meos_mix_reducing(ident1 = "C1", ident2 = "H2", bibref = "Beckmueller et al. (2020) ", beta_v = 1.086000, gamma_v = 0.804000, beta_T = 1.010000, gamma_T = 1.440000)
- type(meos_mix_reducing), parameter **meos_red_508** = meos_mix_reducing(ident1 = "C1", ident2 = "SO2", bibref = "Bell & Herrig (2015)", beta_v = 1.000000000, gamma_v = 1.000000000, beta_T = 1.015760000, gamma_T = 1.084560000)
- type(meos_mix_reducing), parameter **meos_red_509** = meos_mix_reducing(ident1 = "NC4", ident2 = "O2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_510** = meos_mix_reducing(ident1 = "C3_1", ident2 = "R134A", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.32% from 31 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.01623, gamma_T = 0.91325)
- type(meos_mix_reducing), parameter **meos_red_511** = meos_mix_reducing(ident1 = "H2", ident2 = "HE", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_512** = meos_mix_reducing(ident1 = "N2", ident2 = "C3", bibref = "Kunz and Wagner (2007)", beta_v = 0.974424681, gamma_v = 1.081025408, beta_T = 1.002677329, gamma_T = 1.201264026)
- type(meos_mix_reducing), parameter **meos_red_513** = meos_mix_reducing(ident1 = "R1234YF", ident2 = "IC4", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.16% from 49 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.992752903802, gamma_T = 0.93536)
- type(meos_mix_reducing), parameter **meos_red_514** = meos_mix_reducing(ident1 = "ETOH", ident2 = "BENZENE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 0.997008973081, gamma_v = 0.97, beta_T = 0.981354268891, gamma_T = 0.92)
- type(meos_mix_reducing), parameter **meos_red_515** = meos_mix_reducing(ident1 = "CO2", ident2 = "IC4", bibref = "Kunz and Wagner (2007)", beta_v = 1.076551882, gamma_v = 1.081909003, beta_T = 1.023339824, gamma_T = 0.929982936)
- type(meos_mix_reducing), parameter **meos_red_516** = meos_mix_reducing(ident1 = "C1", ident2 = "CYCLOHEX", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.0835, beta_T = 1., gamma_T = 1.3145)
- type(meos_mix_reducing), parameter **meos_red_517** = meos_mix_reducing(ident1 = "N2", ident2 = "MEOH", bibref = "E.W. Lemmon, NIST (2015)", beta_v = 0.534, gamma_v = 1.529, beta_T = 0.998, gamma_T = 1.5293)

- type(meos_mix_reducing), parameter **meos_red_518** = meos_mix_reducing(ident1 = "NC8", ident2 = "O2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_519** = meos_mix_reducing(ident1 = "SO2", ident2 = "MEOH", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.038637307852, gamma_T = 1.0124)
- type(meos_mix_reducing), parameter **meos_red_520** = meos_mix_reducing(ident1 = "C3", ident2 = "CO", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.108143673, beta_T = 1., gamma_T = 1.↩ 197564208)
- type(meos_mix_reducing), parameter **meos_red_521** = meos_mix_reducing(ident1 = "TOLU", ident2 = "NC12", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.82% from 19 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.004944326084, gamma_T = 1.01727)
- type(meos_mix_reducing), parameter **meos_red_522** = meos_mix_reducing(ident1 = "ACETONE", ident2 = "H2O", bibref = "I.H. Bell, NIST (2017) - autofitting code based on data from TDE", beta_v = 1.008091, gamma_v = 0.923029, beta_T = 0.98882, gamma_T = 0.925278)
- type(meos_mix_reducing), parameter **meos_red_523** = meos_mix_reducing(ident1 = "ACETONE", ident2 = "NC9", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.0466, beta_T = 1.0246, gamma_T = 0.9409)
- type(meos_mix_reducing), parameter **meos_red_524** = meos_mix_reducing(ident1 = "NC7", ident2 = "NC9", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.001370076, beta_T = 1., gamma_T = 1.↩ 001150096)
- type(meos_mix_reducing), parameter **meos_red_525** = meos_mix_reducing(ident1 = "CO2", ident2 = "O2", bibref = "Gernert (2013)", beta_v = 1.0000000, gamma_v = 1.0844600, beta_T = 1.000000, gamma_T = 1.0319860)
- type(meos_mix_reducing), parameter **meos_red_526** = meos_mix_reducing(ident1 = "SO2", ident2 = "H2↩ O", bibref = "Koerber and Herrig (2018) / see Herrig (2018) PhD thesis ", beta_v = 1.094032, gamma_v = 0.962547, beta_T = 1.019562, gamma_T = 0.916311)
- type(meos_mix_reducing), parameter **meos_red_527** = meos_mix_reducing(ident1 = "HE", ident2 = "KR", bibref = "E.W. Lemmon, NIST (2017)", beta_v = 1.052, gamma_v = 2.016, beta_T = 0.790, gamma_T = 2.013)
- type(meos_mix_reducing), parameter **meos_red_528** = meos_mix_reducing(ident1 = "F6S", ident2 = "NC5", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.0605, beta_T = 1.↩ 041666666667, gamma_T = 0.9293)
- type(meos_mix_reducing), parameter **meos_red_529** = meos_mix_reducing(ident1 = "O2", ident2 = "CY-CLOHEX", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.3883)
- type(meos_mix_reducing), parameter **meos_red_530** = meos_mix_reducing(ident1 = "IC5", ident2 = "NC6", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.002995876, beta_T = 1., gamma_T = 1.↩ 001204174)
- type(meos_mix_reducing), parameter **meos_red_531** = meos_mix_reducing(ident1 = "SO2", ident2 = "ETOH", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.015537727227, gamma_T = 1.0224)
- type(meos_mix_reducing), parameter **meos_red_532** = meos_mix_reducing(ident1 = "CO2", ident2 = "NC11", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 3.25% from 42 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.0061476, gamma_T = 1.19119)
- type(meos_mix_reducing), parameter **meos_red_533** = meos_mix_reducing(ident1 = "MEOH", ident2 = "NC9", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.118, beta↩ _T = 1.074113856069, gamma_T = 0.893)
- type(meos_mix_reducing), parameter **meos_red_534** = meos_mix_reducing(ident1 = "ETOH", ident2 = "H2O", bibref = "Eckermann and Lemmon (2018) ", beta_v = 1.0124, gamma_v = 0.9558, beta_T = 0.↩ 9866, gamma_T = 0.9971)
- type(meos_mix_reducing), parameter **meos_red_535** = meos_mix_reducing(ident1 = "R23", ident2 = "R114", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0521610, beta_T = 1., gamma_T = 0.9415606)
- type(meos_mix_reducing), parameter **meos_red_536** = meos_mix_reducing(ident1 = "NC7", ident2 = "OXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.27% from 71 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.99904, gamma_T = 0.98672)

- type(meos_mix_reducing), parameter **meos_red_537** = meos_mix_reducing(ident1 = "C3", ident2 = "NC4", bibref = "Kunz and Wagner (2007)", beta_v = 0.999795868, gamma_v = 1.003264179, beta_T = 1.↩ 000310289, gamma_T = 1.007392782)
- type(meos_mix_reducing), parameter **meos_red_538** = meos_mix_reducing(ident1 = "ACETONE", ident2 = "NC11", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010) (estimated from decane and dodecane)", beta_v = 1., gamma_v = 1.075, beta_T = 1.044, gamma_T = 0.98)
- type(meos_mix_reducing), parameter **meos_red_539** = meos_mix_reducing(ident1 = "R23", ident2 = "NC4", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 2% from 37 bubble-point p", beta_v = 1., gamma_v = 1., beta_T = 1.024338277473, gamma_T = 0.89836)
- type(meos_mix_reducing), parameter **meos_red_540** = meos_mix_reducing(ident1 = "R32", ident2 = "R142B", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0305781, beta_T = 1., gamma_T = 0.9718126)
- type(meos_mix_reducing), parameter **meos_red_541** = meos_mix_reducing(ident1 = "ACETONE", ident2 = "MEOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.007, beta_T = 1., gamma_T = 0.97)
- type(meos_mix_reducing), parameter **meos_red_542** = meos_mix_reducing(ident1 = "N2", ident2 = "NC4", bibref = "Kunz and Wagner (2007)", beta_v = 0.99608261, gamma_v = 1.146949309, beta_T = 0.994515234, gamma_T = 1.304886838)
- type(meos_mix_reducing), parameter **meos_red_543** = meos_mix_reducing(ident1 = "C1", ident2 = "N2O", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 3.36% from 49 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.98876, gamma_T = 1.02354)
- type(meos_mix_reducing), parameter **meos_red_544** = meos_mix_reducing(ident1 = "NE", ident2 = "N2", bibref = "E.W. Lemmon, NIST (2016)", beta_v = 0.915750915751, gamma_v = 0.988, beta_T = 1.↩ 025641025641, gamma_T = 1.148)
- type(meos_mix_reducing), parameter **meos_red_545** = meos_mix_reducing(ident1 = "N2", ident2 = "H2", bibref = "Beckmueller et al. (2020) ", beta_v = 0.993000, gamma_v = 0.773000, beta_T = 1.027000, gamma_T = 1.240000)
- type(meos_mix_reducing), parameter **meos_red_546** = meos_mix_reducing(ident1 = "NH3", ident2 = "NC8", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.↩ 231678778175, gamma_T = 1.0676)
- type(meos_mix_reducing), parameter **meos_red_547** = meos_mix_reducing(ident1 = "R134A", ident2 = "ETOH", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 1.↩ 85% from 15 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.029039494536, gamma_T = 0.98183)
- type(meos_mix_reducing), parameter **meos_red_548** = meos_mix_reducing(ident1 = "C1", ident2 = "IC4", bibref = "Kunz and Wagner (2007)", beta_v = 1.011240388, gamma_v = 1.054319053, beta_T = 0.↩ 980315756, gamma_T = 1.161117729)
- type(meos_mix_reducing), parameter **meos_red_549** = meos_mix_reducing(ident1 = "CO2", ident2 = "C3", bibref = "Kunz and Wagner (2007)", beta_v = 0.996898004, gamma_v = 1.047596298, beta_T = 1.↩ 033620538, gamma_T = 0.908772477)
- type(meos_mix_reducing), parameter **meos_red_550** = meos_mix_reducing(ident1 = "CYCLOHEX", ident2 = "PXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.87% from 79 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.00184, gamma_T = 0.99198)
- type(meos_mix_reducing), parameter **meos_red_551** = meos_mix_reducing(ident1 = "KR", ident2 = "NH3", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 2.66% from 8 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.011818, gamma_T = 0.92419)
- type(meos_mix_reducing), parameter **meos_red_552** = meos_mix_reducing(ident1 = "C2", ident2 = "H2O", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_553** = meos_mix_reducing(ident1 = "NC8", ident2 = "AR", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_554** = meos_mix_reducing(ident1 = "EBZN", ident2 = "PXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.38% from 204 bubble-poi", beta_v = 1., gamma_v = 1., beta_T = 0.999479970571, gamma_T = 0.99905)
- type(meos_mix_reducing), parameter **meos_red_555** = meos_mix_reducing(ident1 = "IC4", ident2 = "MEOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1.048, gamma_T = 0.89)
- type(meos_mix_reducing), parameter **meos_red_556** = meos_mix_reducing(ident1 = "NC5", ident2 = "NC12", bibref = "J. Watanasiri and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.047, beta_T = 1., gamma_T = 1.114)

- type(meos_mix_reducing), parameter **meos_red_557** = meos_mix_reducing(ident1 = "P-H2", ident2 = "O-H2", bibref = "ideal mixture", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_558** = meos_mix_reducing(ident1 = "IC5", ident2 = "ACETONE", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 0.9177)
- type(meos_mix_reducing), parameter **meos_red_559** = meos_mix_reducing(ident1 = "ACETONE", ident2 = "NC8", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.0352, beta_T = 1.0196, gamma_T = 0.9277)
- type(meos_mix_reducing), parameter **meos_red_560** = meos_mix_reducing(ident1 = "NH3", ident2 = "H2↩O", bibref = "#Note: The TR1 model shows up first below, but the program uses the last set of parameters,", beta_v = 1.044759, gamma_v = 1.189754, beta_T = 0.933585, gamma_T = 1.015826)
- type(meos_mix_reducing), parameter **meos_red_561** = meos_mix_reducing(ident1 = "NC7", ident2 = "O2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_562** = meos_mix_reducing(ident1 = "IC5", ident2 = "O2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_563** = meos_mix_reducing(ident1 = "H2", ident2 = "NE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2016)", beta_v = 0.8285, gamma_v = 1.2007, beta_T = 1.00705, gamma_T = 0.7819)
- type(meos_mix_reducing), parameter **meos_red_564** = meos_mix_reducing(ident1 = "C2", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.129)
- type(meos_mix_reducing), parameter **meos_red_565** = meos_mix_reducing(ident1 = "CO", ident2 = "MXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 4.78% from 49 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.984390035020, gamma_T = 1.5975)
- type(meos_mix_reducing), parameter **meos_red_566** = meos_mix_reducing(ident1 = "NC7", ident2 = "NC12", bibref = "J. Watanasiri and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.02, beta_T = 1., gamma_T = 1.019)
- type(meos_mix_reducing), parameter **meos_red_567** = meos_mix_reducing(ident1 = "C1", ident2 = "CO2", bibref = "Kunz and Wagner (2007)", beta_v = 0.999518072, gamma_v = 1.002806594, beta_T = 1.02262449, gamma_T = 0.975665369)
- type(meos_mix_reducing), parameter **meos_red_568** = meos_mix_reducing(ident1 = "NC7", ident2 = "H2O", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_569** = meos_mix_reducing(ident1 = "R32", ident2 = "R1234YF", bibref = "R. Akasaka, FPE, 2013, DOI:10.1016/j.fluid.2013.07.057", beta_v = 0.993346, gamma_v = 1.02211, beta_T = 1.00052, gamma_T = 0.948538)
- type(meos_mix_reducing), parameter **meos_red_570** = meos_mix_reducing(ident1 = "NC8", ident2 = "NC10", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1.002553544, beta_T = 1., gamma_T = 1.007186267)
- type(meos_mix_reducing), parameter **meos_red_571** = meos_mix_reducing(ident1 = "H2S", ident2 = "HE", bibref = "Kunz and Wagner (2012)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_572** = meos_mix_reducing(ident1 = "NC8", ident2 = "H2", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.305249405, beta_T = 1., gamma_T = 2.↩191555216)
- type(meos_mix_reducing), parameter **meos_red_573** = meos_mix_reducing(ident1 = "C1", ident2 = "NC12", bibref = "J. Watanasiri and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 0.978, beta_T = 0.904, gamma_T = 1.716)
- type(meos_mix_reducing), parameter **meos_red_574** = meos_mix_reducing(ident1 = "CO2", ident2 = "C2", bibref = "Kunz and Wagner (2007)", beta_v = 1.002525718, gamma_v = 1.032876701, beta_T = 1.↩013871147, gamma_T = 0.90094953)
- type(meos_mix_reducing), parameter **meos_red_575** = meos_mix_reducing(ident1 = "NC5", ident2 = "HE", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_576** = meos_mix_reducing(ident1 = "IC5", ident2 = "NC7", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.009928206, beta_T = 1., gamma_T = 1.↩003194615)
- type(meos_mix_reducing), parameter **meos_red_577** = meos_mix_reducing(ident1 = "H2S", ident2 = "MEG", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.9877, gamma_T = 1.1981)

- type(meos_mix_reducing), parameter **meos_red_578** = meos_mix_reducing(ident1 = "R21", ident2 = "MEOH", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.74% from 4 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 0.969208253777, gamma_T = 0.91282)
- type(meos_mix_reducing), parameter **meos_red_579** = meos_mix_reducing(ident1 = "N2", ident2 = "IC5", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.154135439, beta_T = 1., gamma_T = 1.↩ 38177077)
- type(meos_mix_reducing), parameter **meos_red_580** = meos_mix_reducing(ident1 = "CO2", ident2 = "CYCLOHEX", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1.19, gamma_v = 1.0106, beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_581** = meos_mix_reducing(ident1 = "TOLU", ident2 = "OXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 1.↩ 16% from 67 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.001743, gamma_T = 1.00209)
- type(meos_mix_reducing), parameter **meos_red_582** = meos_mix_reducing(ident1 = "N2", ident2 = "R12", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0625855, beta_T = 1., gamma_T = 1.↩ 2981559)
- type(meos_mix_reducing), parameter **meos_red_583** = meos_mix_reducing(ident1 = "NC5", ident2 = "NC8", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.069223964, beta_T = 1., gamma_T = 1.↩ 016422347)
- type(meos_mix_reducing), parameter **meos_red_584** = meos_mix_reducing(ident1 = "PXYL", ident2 = "NC11", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.14% from 12 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.000190036107, gamma_T = 1.0018)
- type(meos_mix_reducing), parameter **meos_red_585** = meos_mix_reducing(ident1 = "H2S", ident2 = "MEOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.042)
- type(meos_mix_reducing), parameter **meos_red_586** = meos_mix_reducing(ident1 = "R23", ident2 = "ETOH", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 2.5% from 9 bubble-point ", beta_v = 1., gamma_v = 1., beta_T = 1.104764850802, gamma_T = 1.05138)
- type(meos_mix_reducing), parameter **meos_red_587** = meos_mix_reducing(ident1 = "R1234YF", ident2 = "R134A", bibref = "E.W. Lemmon, NIST (2013)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 0.985)
- type(meos_mix_reducing), parameter **meos_red_588** = meos_mix_reducing(ident1 = "ACETONE", ident2 = "ETOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 0.997, beta_T = 1.005, gamma_T = 0.975)
- type(meos_mix_reducing), parameter **meos_red_589** = meos_mix_reducing(ident1 = "NC4", ident2 = "BENZENE", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_590** = meos_mix_reducing(ident1 = "R1234ZE", ident2 = "NC5", bibref = "S.L. Outcalt and E.W. Lemmon, NIST (2012)", beta_v = 1., gamma_v = 1., beta_T = 1.↩ 007049345418, gamma_T = 0.937)
- type(meos_mix_reducing), parameter **meos_red_591** = meos_mix_reducing(ident1 = "C1", ident2 = "CO", bibref = "Kunz and Wagner (2007)", beta_v = 0.997340772, gamma_v = 1.006102927, beta_T = 0.↩ 987411732, gamma_T = 0.987473033)
- type(meos_mix_reducing), parameter **meos_red_592** = meos_mix_reducing(ident1 = "CO2", ident2 = "H2↩ O", bibref = "Gernert (2013)", beta_v = 1.0213920, gamma_v = 0.8951560, beta_T = 1.030538, gamma_T = 0.8284720)
- type(meos_mix_reducing), parameter **meos_red_593** = meos_mix_reducing(ident1 = "CO2", ident2 = "IC5", bibref = "Kunz and Wagner (2007)", beta_v = 1.060793104, gamma_v = 1.116793198, beta_T = 1.↩ 019180957, gamma_T = 0.961218039)
- type(meos_mix_reducing), parameter **meos_red_594** = meos_mix_reducing(ident1 = "NC5", ident2 = "ETOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.025, beta_T = 1.028, gamma_T = 0.923)
- type(meos_mix_reducing), parameter **meos_red_595** = meos_mix_reducing(ident1 = "XE", ident2 = "F6S", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.61% from 81 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.0366024, gamma_T = 0.92846)
- type(meos_mix_reducing), parameter **meos_red_596** = meos_mix_reducing(ident1 = "C2", ident2 = "MEOH", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 0.977, gamma_T = 1.011)

- type(meos_mix_reducing), parameter **meos_red_597** = meos_mix_reducing(ident1 = "XE", ident2 = "C3", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.41% from 22 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 0.995113990308, gamma_T = 1.04133)
- type(meos_mix_reducing), parameter **meos_red_598** = meos_mix_reducing(ident1 = "NC5", ident2 = "H2S", bibref = "Kunz and Wagner (2012)", beta_v = 0.984613203, gamma_v = 1.076539234, beta_T = 0.962006651, gamma_T = 0.959065662)
- type(meos_mix_reducing), parameter **meos_red_599** = meos_mix_reducing(ident1 = "SO2", ident2 = "H2S", bibref = "Herrig (2018) - Lorentz-Berthelot Combining Rules / see Herrig (2018) PhD thesis", beta_v = 1.000000000, gamma_v = 1.000000000, beta_T = 1.000000000, gamma_T = 1.000000000)
- type(meos_mix_reducing), parameter **meos_red_600** = meos_mix_reducing(ident1 = "KR", ident2 = "XE", bibref = "E.W. Lemmon, NIST (2002)", beta_v = 1., gamma_v = 1.0054519, beta_T = 1., gamma_T = 1.0196370)
- type(meos_mix_reducing), parameter **meos_red_601** = meos_mix_reducing(ident1 = "CO", ident2 = "H2S", bibref = "Kunz and Wagner (2012)", beta_v = 0.795660392, gamma_v = 1.101731308, beta_T = 1.025536736, gamma_T = 1.022749748)
- type(meos_mix_reducing), parameter **meos_red_602** = meos_mix_reducing(ident1 = "NH3", ident2 = "H2S", bibref = "Neumann et al. (2020)", beta_v = 1.000000, gamma_v = 1.000000, beta_T = 1.000000, gamma_T = 1.000000)
- type(meos_mix_reducing), parameter **meos_red_603** = meos_mix_reducing(ident1 = "PRLN", ident2 = "ACETONE", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 2.4% from 33 bubble-point", beta_v = 1., gamma_v = 1., beta_T = 1.019919, gamma_T = 0.99014)
- type(meos_mix_reducing), parameter **meos_red_604** = meos_mix_reducing(ident1 = "O2", ident2 = "SO2", bibref = "Bell & Herrig (2015)", beta_v = 1.000000000, gamma_v = 1.000000000, beta_T = 0.991148000, gamma_T = 1.173450000)
- type(meos_mix_reducing), parameter **meos_red_605** = meos_mix_reducing(ident1 = "CO2", ident2 = "NC4", bibref = "Kunz and Wagner (2007)", beta_v = 1.174760923, gamma_v = 1.222437324, beta_T = 1.018171004, gamma_T = 0.911498231)
- type(meos_mix_reducing), parameter **meos_red_606** = meos_mix_reducing(ident1 = "MEOH", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1.183, beta_T = 0.947867298578, gamma_T = 0.764)
- type(meos_mix_reducing), parameter **meos_red_607** = meos_mix_reducing(ident1 = "NC5", ident2 = "CYCLOHEX", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 0.9961)
- type(meos_mix_reducing), parameter **meos_red_608** = meos_mix_reducing(ident1 = "NC7", ident2 = "PXYL", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.25% from 124 bubble-poi", beta_v = 1., gamma_v = 1., beta_T = 0.99936, gamma_T = 0.98957)
- type(meos_mix_reducing), parameter **meos_red_609** = meos_mix_reducing(ident1 = "IC4", ident2 = "H2O", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_610** = meos_mix_reducing(ident1 = "IC5", ident2 = "NC12", bibref = "J. Watanasiri and E.W. Lemmon, NIST (2010) (estimated from propane and pentane mixed with dodecane)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.13)
- type(meos_mix_reducing), parameter **meos_red_611** = meos_mix_reducing(ident1 = "C2", ident2 = "CO", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1.201417898, beta_T = 1., gamma_T = 1.069224728)
- type(meos_mix_reducing), parameter **meos_red_612** = meos_mix_reducing(ident1 = "C3", ident2 = "SO2", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 0.8738)
- type(meos_mix_reducing), parameter **meos_red_613** = meos_mix_reducing(ident1 = "PRLN", ident2 = "NC5", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010) (estimated from trend found in C1-C10)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.04)
- type(meos_mix_reducing), parameter **meos_red_614** = meos_mix_reducing(ident1 = "C3", ident2 = "TOLU", bibref = "T.M. Blackham and E.W. Lemmon, NIST (2010)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.049)
- type(meos_mix_reducing), parameter **meos_red_615** = meos_mix_reducing(ident1 = "NC8", ident2 = "HE", bibref = "Kunz and Wagner (2007)", beta_v = 1., gamma_v = 1., beta_T = 1., gamma_T = 1.)
- type(meos_mix_reducing), parameter **meos_red_616** = meos_mix_reducing(ident1 = "NC6", ident2 = "EBZN", bibref = "I.H. Bell and E.W. Lemmon, JCED, 2016. DOI:10.1021/acs.jced.6b00257; MAPD: 0.09% from 33 bubble-poin", beta_v = 1., gamma_v = 1., beta_T = 1.0002601, gamma_T = 0.99383)

- type(meos_mix_reducing), parameter **meos_red_617** = meos_mix_reducing(ident1 = "CYCLOHEX", ident2 = "NC10", bibref = "I. Cullimore and E.W. Lemmon, NIST (2010) (estimated from octane/cyclohexane and dodecane/cyclohexan", beta_v = 1., gamma_v = 1.012, beta_T = 1., gamma_T = 1.025)
- integer, parameter **max_meos_mix_reducing** = 617
- type(meos_mix_reducing), dimension(max_meos_mix_reducing), parameter **meos_mix_reducingdb** = (/ meos_red_1,meos_red_2,meos_red_3,meos_red_4,meos_red_5,meos_red_6, meos_red_7,meos_↩ red_8,meos_red_9,meos_red_10,meos_red_11,meos_red_12, meos_red_13,meos_red_14,meos_red↩ _15,meos_red_16,meos_red_17,meos_red_18, meos_red_19,meos_red_20,meos_red_21,meos_red_↩ 22,meos_red_23,meos_red_24, meos_red_25,meos_red_26,meos_red_27,meos_red_28,meos_red_↩ 29,meos_red_30, meos_red_31,meos_red_32,meos_red_33,meos_red_34,meos_red_35,meos_red_↩ 36, meos_red_37,meos_red_38,meos_red_39,meos_red_40,meos_red_41,meos_red_42, meos_red_↩ 43,meos_red_44,meos_red_45,meos_red_46,meos_red_47,meos_red_48, meos_red_49,meos_red_↩ 50,meos_red_51,meos_red_52,meos_red_53,meos_red_54, meos_red_55,meos_red_56,meos_red_↩ 57,meos_red_58,meos_red_59,meos_red_60, meos_red_61,meos_red_62,meos_red_63,meos_red_↩ 64,meos_red_65,meos_red_66, meos_red_67,meos_red_68,meos_red_69,meos_red_70,meos_red_↩ 71,meos_red_72, meos_red_73,meos_red_74,meos_red_75,meos_red_76,meos_red_77,meos_red_↩ 78, meos_red_79,meos_red_80,meos_red_81,meos_red_82,meos_red_83,meos_red_84, meos_red_↩ 85,meos_red_86,meos_red_87,meos_red_88,meos_red_89,meos_red_90, meos_red_91,meos_red_↩ 92,meos_red_93,meos_red_94,meos_red_95,meos_red_96, meos_red_97,meos_red_98,meos_red_↩ 99,meos_red_100,meos_red_101,meos_red_102, meos_red_103,meos_red_104,meos_red_105,meos↩ _red_106,meos_red_107,meos_red_108, meos_red_109,meos_red_110,meos_red_111,meos_red_↩ 112,meos_red_113,meos_red_114, meos_red_115,meos_red_116,meos_red_117,meos_red_118,meos↩ _red_119,meos_red_120, meos_red_121,meos_red_122,meos_red_123,meos_red_124,meos_red_↩ 125,meos_red_126, meos_red_127,meos_red_128,meos_red_129,meos_red_130,meos_red_131,meos↩ _red_132, meos_red_133,meos_red_134,meos_red_135,meos_red_136,meos_red_137,meos_red_138, meos_red_139,meos_red_140,meos_red_141,meos_red_142,meos_red_143,meos_red_144, meos↩ _red_145,meos_red_146,meos_red_147,meos_red_148,meos_red_149,meos_red_150, meos_red_↩ 151,meos_red_152,meos_red_153,meos_red_154,meos_red_155,meos_red_156, meos_red_157,meos↩ _red_158,meos_red_159,meos_red_160,meos_red_161,meos_red_162, meos_red_163,meos_red_↩ 164,meos_red_165,meos_red_166,meos_red_167,meos_red_168, meos_red_169,meos_red_170,meos↩ _red_171,meos_red_172,meos_red_173,meos_red_174, meos_red_175,meos_red_176,meos_red_↩ 177,meos_red_178,meos_red_179,meos_red_180, meos_red_181,meos_red_182,meos_red_183,meos↩ _red_184,meos_red_185,meos_red_186, meos_red_187,meos_red_188,meos_red_189,meos_red_↩ 190,meos_red_191,meos_red_192, meos_red_193,meos_red_194,meos_red_195,meos_red_196,meos↩ _red_197,meos_red_198, meos_red_199,meos_red_200,meos_red_201,meos_red_202,meos_red_↩ 203,meos_red_204, meos_red_205,meos_red_206,meos_red_207,meos_red_208,meos_red_209,meos↩ _red_210, meos_red_211,meos_red_212,meos_red_213,meos_red_214,meos_red_215,meos_red_216, meos_red_217,meos_red_218,meos_red_219,meos_red_220,meos_red_221,meos_red_222, meos↩ _red_223,meos_red_224,meos_red_225,meos_red_226,meos_red_227,meos_red_228, meos_red_↩ 229,meos_red_230,meos_red_231,meos_red_232,meos_red_233,meos_red_234, meos_red_235,meos↩ _red_236,meos_red_237,meos_red_238,meos_red_239,meos_red_240, meos_red_241,meos_red_↩ 242,meos_red_243,meos_red_244,meos_red_245,meos_red_246, meos_red_247,meos_red_248,meos↩ _red_249,meos_red_250,meos_red_251,meos_red_252, meos_red_253,meos_red_254,meos_red_↩ 255,meos_red_256,meos_red_257,meos_red_258, meos_red_259,meos_red_260,meos_red_261,meos↩ _red_262,meos_red_263,meos_red_264, meos_red_265,meos_red_266,meos_red_267,meos_red_↩ 268,meos_red_269,meos_red_270, meos_red_271,meos_red_272,meos_red_273,meos_red_274,meos↩ _red_275,meos_red_276, meos_red_277,meos_red_278,meos_red_279,meos_red_280,meos_red_↩ 281,meos_red_282, meos_red_283,meos_red_284,meos_red_285,meos_red_286,meos_red_287,meos↩ _red_288, meos_red_289,meos_red_290,meos_red_291,meos_red_292,meos_red_293,meos_red_294, meos_red_295,meos_red_296,meos_red_297,meos_red_298,meos_red_299,meos_red_300, meos↩ _red_301,meos_red_302,meos_red_303,meos_red_304,meos_red_305,meos_red_306, meos_red_↩ 307,meos_red_308,meos_red_309,meos_red_310,meos_red_311,meos_red_312, meos_red_313,meos↩ _red_314,meos_red_315,meos_red_316,meos_red_317,meos_red_318, meos_red_319,meos_red_↩ 320,meos_red_321,meos_red_322,meos_red_323,meos_red_324, meos_red_325,meos_red_326,meos↩ _red_327,meos_red_328,meos_red_329,meos_red_330, meos_red_331,meos_red_332,meos_red_↩ 333,meos_red_334,meos_red_335,meos_red_336, meos_red_337,meos_red_338,meos_red_339,meos↩ _red_340,meos_red_341,meos_red_342, meos_red_343,meos_red_344,meos_red_345,meos_red_↩ 346,meos_red_347,meos_red_348, meos_red_349,meos_red_350,meos_red_351,meos_red_352,meos↩

_red_353,meos_red_354, meos_red_355,meos_red_356,meos_red_357,meos_red_358,meos_red_↩
359,meos_red_360, meos_red_361,meos_red_362,meos_red_363,meos_red_364,meos_red_365,meos↩
_red_366, meos_red_367,meos_red_368,meos_red_369,meos_red_370,meos_red_371,meos_red_372,
meos_red_373,meos_red_374,meos_red_375,meos_red_376,meos_red_377,meos_red_378, meos↩
_red_379,meos_red_380,meos_red_381,meos_red_382,meos_red_383,meos_red_384, meos_red_↩
385,meos_red_386,meos_red_387,meos_red_388,meos_red_389,meos_red_390, meos_red_391,meos↩
_red_392,meos_red_393,meos_red_394,meos_red_395,meos_red_396, meos_red_397,meos_red_↩
398,meos_red_399,meos_red_400,meos_red_401,meos_red_402, meos_red_403,meos_red_404,meos↩
_red_405,meos_red_406,meos_red_407,meos_red_408, meos_red_409,meos_red_410,meos_red_↩
411,meos_red_412,meos_red_413,meos_red_414, meos_red_415,meos_red_416,meos_red_417,meos↩
_red_418,meos_red_419,meos_red_420, meos_red_421,meos_red_422,meos_red_423,meos_red_↩
424,meos_red_425,meos_red_426, meos_red_427,meos_red_428,meos_red_429,meos_red_430,meos↩
_red_431,meos_red_432, meos_red_433,meos_red_434,meos_red_435,meos_red_436,meos_red_↩
437,meos_red_438, meos_red_439,meos_red_440,meos_red_441,meos_red_442,meos_red_443,meos↩
_red_444, meos_red_445,meos_red_446,meos_red_447,meos_red_448,meos_red_449,meos_red_450,
meos_red_451,meos_red_452,meos_red_453,meos_red_454,meos_red_455,meos_red_456, meos↩
_red_457,meos_red_458,meos_red_459,meos_red_460,meos_red_461,meos_red_462, meos_red_↩
463,meos_red_464,meos_red_465,meos_red_466,meos_red_467,meos_red_468, meos_red_469,meos↩
_red_470,meos_red_471,meos_red_472,meos_red_473,meos_red_474, meos_red_475,meos_red_↩
476,meos_red_477,meos_red_478,meos_red_479,meos_red_480, meos_red_481,meos_red_482,meos↩
_red_483,meos_red_484,meos_red_485,meos_red_486, meos_red_487,meos_red_488,meos_red_↩
489,meos_red_490,meos_red_491,meos_red_492, meos_red_493,meos_red_494,meos_red_495,meos↩
_red_496,meos_red_497,meos_red_498, meos_red_499,meos_red_500,meos_red_501,meos_red_↩
502,meos_red_503,meos_red_504, meos_red_505,meos_red_506,meos_red_507,meos_red_508,meos↩
_red_509,meos_red_510, meos_red_511,meos_red_512,meos_red_513,meos_red_514,meos_red_↩
515,meos_red_516, meos_red_517,meos_red_518,meos_red_519,meos_red_520,meos_red_521,meos↩
_red_522, meos_red_523,meos_red_524,meos_red_525,meos_red_526,meos_red_527,meos_red_528,
meos_red_529,meos_red_530,meos_red_531,meos_red_532,meos_red_533,meos_red_534, meos↩
_red_535,meos_red_536,meos_red_537,meos_red_538,meos_red_539,meos_red_540, meos_red_↩
541,meos_red_542,meos_red_543,meos_red_544,meos_red_545,meos_red_546, meos_red_547,meos↩
_red_548,meos_red_549,meos_red_550,meos_red_551,meos_red_552, meos_red_553,meos_red_↩
554,meos_red_555,meos_red_556,meos_red_557,meos_red_558, meos_red_559,meos_red_560,meos↩
_red_561,meos_red_562,meos_red_563,meos_red_564, meos_red_565,meos_red_566,meos_red_↩
567,meos_red_568,meos_red_569,meos_red_570, meos_red_571,meos_red_572,meos_red_573,meos↩
_red_574,meos_red_575,meos_red_576, meos_red_577,meos_red_578,meos_red_579,meos_red_↩
580,meos_red_581,meos_red_582, meos_red_583,meos_red_584,meos_red_585,meos_red_586,meos↩
_red_587,meos_red_588, meos_red_589,meos_red_590,meos_red_591,meos_red_592,meos_red_↩
593,meos_red_594, meos_red_595,meos_red_596,meos_red_597,meos_red_598,meos_red_599,meos↩
_red_600, meos_red_601,meos_red_602,meos_red_603,meos_red_604,meos_red_605,meos_red_606,
meos_red_607,meos_red_608,meos_red_609,meos_red_610,meos_red_611,meos_red_612, meos_red↩
_613,meos_red_614,meos_red_615,meos_red_616,meos_red_617 /)

- type(meos_mix_data), parameter **meos_mix1** = meos_mix_data(ident1 = "H2", ident2 = "NH3", bibref = "", Fij = 1.000000, num_mix = 4, n_mix = (/ -3.73558,-7.47092,1.98413, 1.87191,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 1.2800000000,2.0500000000,2.6000000000, 3.1300000000,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,2,1,2,0,0, 0,0,0,0,0,0 /), l_mix = (/ 1,1,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,-0.6100000000, -1.6000000000,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.8000000000, 1.6200000000,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,-2.0600000000, -1.7400000000,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.7900000000, 2.1000000000,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 2, num_gauss = 2)
- type(meos_mix_data), parameter **meos_mix2** = meos_mix_data(ident1 = "N2", ident2 = "BENZENE", bibref = "", Fij = 0.257, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩
11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩
0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩

0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)

- type(meos_mix_data), parameter **meos_mix3** = meos_mix_data(ident1 = "CO2", ident2 = "H2", bibref = "", Fij = 1.000000, num_mix = 6, n_mix = (/ 0.356000000000D+01,-0.103600000000D+01,-0.483500000000D+01, 0.123800000000D+02,-0.265000000000D+01,-0.330000000000D+01, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 1.4700000000,1.1700000000,1.9500000000, 0.3100000000,1.4120000000,2.2800000000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,2,1,2,3,1, 0,0,0,0,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,-0.5800000000, -0.2000000000,-0.2920000000,-0.1200000000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.5200000000, 0.1500000000,0.2400000000,0.1500000000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,-0.4650000000, -0.8200000000,-0.5200000000,-1.0000000000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.1700000000, 2.1100000000,1.4900000000,1.7300000000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 4)
- type(meos_mix_data), parameter **meos_mix4** = meos_mix_data(ident1 = "C1", ident2 = "ETOH", bibref = "", Fij = 1.486, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix5** = meos_mix_data(ident1 = "MEOH", ident2 = "NC7", bibref = "", Fij = 9.577000, num_mix = 4, n_mix = (/ -0.013073,0.018259,0.81299e-5, 0.0078496,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 7.4,0.35,10.0, 5.3,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,3,11,2,0,0, 0,0,0,0,0,0 /), l_mix = (/ 1,1,2,3,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 4, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix6** = meos_mix_data(ident1 = "C2", ident2 = "C3", bibref = "", Fij = 0.13042476515, num_mix = 10, n_mix = (/ 0.25574776844118d1,-0.79846357136353d1,0.47859131465806d1, -0.73265392369587,0.13805471345312d1,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.55527385721943d-4,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix7** = meos_mix_data(ident1 = "TOLU", ident2 = "NC9", bibref = "", Fij = -1.06, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix8** = meos_mix_data(ident1 = "NC5", ident2 = "MEOH", bibref = "", Fij = 10.487000, num_mix = 4, n_mix = (/ -0.013073,0.018259,0.81299e-5, 0.0078496,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 7.4,0.35,10.0, 5.3,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,3,11,2,0,0, 0,0,0,0,0,0 /), l_mix = (/ 1,1,2,3,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 4, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix9** = meos_mix_data(ident1 = "CO2", ident2 = "R1234YF", bibref = "", Fij = -0.657, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806,

-0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩
11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩
0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0,
0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)

- type(meos_mix_data), parameter **meos_mix10** = meos_mix_data(ident1 = "NH3", ident2 = "CYCLOHEX", bibref = "", Fij = -0.08997, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩ 0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix11** = meos_mix_data(ident1 = "NC10", ident2 = "NC12", bibref = "", Fij = 0.35, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩ 0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix12** = meos_mix_data(ident1 = "C2", ident2 = "IC4", bibref = "", Fij = 0.260632376098, num_mix = 10, n_mix = (/ 0.25574776844118d1,-0.79846357136353d1,0.↩ 47859131465806d1, -0.73265392369587,0.13805471345312d1,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.55527385721943d-4,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.↩ 25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_↩ mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix13** = meos_mix_data(ident1 = "NE", ident2 = "XE", bibref = "", Fij = 1., num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.↩ 73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.↩ 25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix14** = meos_mix_data(ident1 = "ETOH", ident2 = "CYCLOHEX", bibref = "", Fij = 1., num_mix = 7, n_mix = (/ -2.05560,1.94512,-0.196687, 0.425412,-1.01883,-0.786737, -0.229943,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 4.5,4.7,4.68, 2.5,4.68,5.5, 8.2,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0 /), d_mix = (/ 1,1,2,3,4,4, 5,0,0,0,0,0 /), l_mix = (/ 0,0,0,1,1,2, 2,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 5, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix15** = meos_mix_data(ident1 = "CO", ident2 = "H2", bibref = "", Fij = 1.000000, num_mix = 4, n_mix = (/ -0.521000000000D+00,-0.387000000000D+00,-0.259000000000D+01, 0.435000000000D+01,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 2.2500000000,0.↩ 4730000000,0.5850000000, 0.0910000000,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,2,1,2,0,0, 0,0,0,0,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,-0.6470000000, -0.3440000000,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,1.↩

3800000000, 0.7730000000,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.↩
0d0,0.0d0,-0.7510000000, -0.6600000000,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix
= (/ 0.0d0,0.0d0,1.8600000000, 2.2300000000,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num↩
_exp = 0, num_gauss = 2)

- type(meos_mix_data), parameter **meos_mix16** = meos_mix_data(ident1 = "N2", ident2 = "ETOH", bibref = "", Fij = 0.778, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩
11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩
0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix17** = meos_mix_data(ident1 = "CO2", ident2 = "R41", bibref = "", Fij = -0.5483, num_mix = 4, n_mix = (/ -0.013073,0.018259,0.81299e-5, 0.0078496,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 7.4,0.35,10.0, 5.3,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), d_mix = (/ 1,3,11,2,0,0, 0,0,0,0,0,0 /), l_mix = (/ 1,1,2,3,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 4, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix18** = meos_mix_data(ident1 = "CO2", ident2 = "BENZENE", bibref = "", Fij = 2.894, num_mix = 9, n_mix = (/ 0.013746429958576,-0.0074425012129552,-0.↩
0045516600213685, -0.0054546603350237,0.0023682016824471,0.18007763721438, -0.44773942932486,0.↩
019327374888200,-0.30632197804624, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 1.85,3.95,0.0, 1.85,3.85,5.25, 3.↩
85,0.2,6.5, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 3,3,4,4,4,1, 1,1,2,0,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.25, 0.25,0.0,0.0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.5, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.5, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.75, 1.0,2.0,3.0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 4)
- type(meos_mix_data), parameter **meos_mix19** = meos_mix_data(ident1 = "ACETONE", ident2 = "CYCLOHEX", bibref = "", Fij = -0.5274, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.↩
7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.↩
10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon↩
_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix20** = meos_mix_data(ident1 = "C1", ident2 = "NC4", bibref = "", Fij = 1., num_mix = 10, n_mix = (/ 0.25574776844118d1,-0.79846357136353d1,0.47859131465806d1, -0.↩
73265392369587,0.13805471345312d1,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩
11836314681968, 0.55527385721943d-4,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix21** = meos_mix_data(ident1 = "PRLN", ident2 = "H2O", bibref = "", Fij = 0.7604, num_mix = 6, n_mix = (/ 1.09765,1.94679,-2.16809, -0.137077,0.0486690,1.04024, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 0.26,7.3,5.3, 2.3,0.7,3.3, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), d_mix = (/ 2,3,5,5,7,6, 0,0,0,0,0,0 /), l_mix = (/ 1,2,2,1,1,2, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 6, num_gauss = 0)

- type(meos_mix_data), parameter **meos_mix22** = meos_mix_data(ident1 = "CO2", ident2 = "PRLN", bibref = "", Fij = -0.362, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩ 0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix23** = meos_mix_data(ident1 = "C1", ident2 = "N2", bibref = "", Fij = 1., num_mix = 9, n_mix = (/ -0.98038985517335d-2,0.42487270143005d-3,-0.34800214576142d-1, -0.↩ 13333813013896,-0.11993694974627d-1,0.69243379775168d-1, -0.31022508148249,0.24495491753226,0.↩ 22369816716981, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 0.0,1.85,7.85, 5.4,0.0,0.75, 2.8,4.45,4.25, 0.0d0,0.↩ 0d0,0.0d0 /), d_mix = (/ 1,4,1,2,2,2, 2,2,3,0,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.↩ 0d0,1.0d0, 1.0d0,0.25,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.5, 0.5,0.5,0.5, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.5, 0.5,0.5,0.5, 0.5,0.5,0.5, 0.↩ 0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,1.0d0, 1.0d0,2.5,3.0d0, 3.0d0,3.0d0,3.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 7)
- type(meos_mix_data), parameter **meos_mix24** = meos_mix_data(ident1 = "NC4", ident2 = "IC4", bibref = "", Fij = -0.0551240293009, num_mix = 10, n_mix = (/ 0.25574776844118d1,-0.79846357136353d1,0.↩ 47859131465806d1, -0.73265392369587,0.13805471345312d1,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.55527385721943d-4,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.↩ 25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_↩ mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix25** = meos_mix_data(ident1 = "CYCLOHEX", ident2 = "NC12", bibref = "", Fij = -0.358, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩ 0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix26** = meos_mix_data(ident1 = "AR", ident2 = "NH3", bibref = "", Fij = 1., num_mix = 3, n_mix = (/ 0.02350785,-1.913776,1.624062, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 2.3,1.65,0.42, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 3,1,1,0,0,0, 0,0,0,0,0,0 /), l_mix = (/ 1,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,-1.3,-1.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.31,0.39, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,-0.6,-0.5, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.9,1.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 1, num_gauss = 2)
- type(meos_mix_data), parameter **meos_mix27** = meos_mix_data(ident1 = "N2", ident2 = "C2", bibref = "", Fij = 1., num_mix = 6, n_mix = (/ -0.47376518126608,0.48961193461001,-0.57011062090535d-2, -0.↩ 19966820041320,-0.69411103101723,0.69226192739021, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 0.0,0.05,0.0, 3.65,4.9,4.45, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 2,2,3,1,2,2, 0,0,0,0,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 1.0d0,1.0d0,0.875, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 1.0d0,1.0d0,1.25, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 3)
- type(meos_mix_data), parameter **meos_mix28** = meos_mix_data(ident1 = "CYCLOHEX", ident2 = "NC9", bibref = "", Fij = -0.358, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩

0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)

- type(meos_mix_data), parameter **meos_mix29** = meos_mix_data(ident1 = "BENZENE", ident2 = "TOLU", bibref = "", Fij = 0.068, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix30** = meos_mix_data(ident1 = "NC7", ident2 = "TOLU", bibref = "", Fij = -0.6281548, num_mix = 12, n_mix = (/ -0.00080926050298746,-0.00075381925080059,-0.041618768891219, -0.23452173681569,0.14003840584586,0.063281744807738, -0.034660425848809,-0.23918747334251,0.0019855255066891, 6.1777746171555,-6.9575358271105,1.0630185306388 /), t_mix = (/ 0.65,1.55,3.1, 5.9,7.05,3.35, 1.2,5.8,2.7, 0.45,0.55,1.95 /), d_mix = (/ 3,4,1,2,2,2, 2,2,2,3,3,3 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,1.0, 1.0,1.0,0.875, 0.75,0.5,0.0, 0.0,0.0,0.0 /), gamma_mix = (/ 0.0d0,0.0d0,0.5, 0.5,0.5,0.5, 0.5,0.5,0.5, 0.5,0.5,0.5 /), epsilon_mix = (/ 0.0d0,0.0d0,0.5, 0.5,0.5,0.5, 0.5,0.5,0.5, 0.5,0.5,0.5 /), beta_mix = (/ 0.0d0,0.0d0,1.0, 1.0,1.0,1.25, 1.5,2.0,3.0, 3.0,3.0,3.0 /), num_exp = 0, num_gauss = 10)
- type(meos_mix_data), parameter **meos_mix31** = meos_mix_data(ident1 = "C2", ident2 = "NC4", bibref = "", Fij = 0.281570073085, num_mix = 10, n_mix = (/ 0.25574776844118d1,-0.79846357136353d1,0.47859131465806d1, -0.73265392369587,0.13805471345312d1,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.55527385721943d-4,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix32** = meos_mix_data(ident1 = "N2", ident2 = "NC12", bibref = "", Fij = 0.898, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix33** = meos_mix_data(ident1 = "CO2", ident2 = "TOLU", bibref = "", Fij = 1.257, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix34** = meos_mix_data(ident1 = "TOLU", ident2 = "NC8", bibref = "", Fij = -0.704, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0,

0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)

- type(meos_mix_data), parameter **meos_mix35** = meos_mix_data(ident1 = "R143A", ident2 = "R134A", bibref = "", Fij = 0.5557, num_mix = 4, n_mix = (/ -0.013073,0.018259,0.81299e-5, 0.0078496,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 7.4,0.35,10.0, 5.3,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0 /), d_mix = (/ 1,3,11,2,0,0, 0,0,0,0,0,0 /), l_mix = (/ 1,1,2,3,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 4, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix36** = meos_mix_data(ident1 = "HE", ident2 = "NE", bibref = "", Fij = -3.25, num_mix = 6, n_mix = (/ -0.47376518126608,0.48961193461001,-0.0057011062090535, -0.↩
19966820041320,-0.69411103101723,0.69226192739021, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 0.0,0.05,0.0, 3.65,4.9,4.45, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 2,2,3,1,2,2, 0,0,0,0,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 1.0,1.0,0.875, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 1.0,1.0,1.25, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 3)
- type(meos_mix_data), parameter **meos_mix37** = meos_mix_data(ident1 = "CYCLOHEX", ident2 = "NC11", bibref = "", Fij = -0.358, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩
11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩
0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix38** = meos_mix_data(ident1 = "CO", ident2 = "H2O", bibref = "", Fij = 0.9897000, num_mix = 5, n_mix = (/ 0.401420790000D+01,-0.115739390000D+01,-0.↩
721024250000D+01, -0.532512230000D+01,-0.221558670000D+01,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), t_mix = (/ 0.5470000000,0.0550000000,1.9250000000, 0.5520000000,1.0000000000,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,4,0, 0,0,0,0,0,0 /), l_mix = (/ 0,1,1,1,1,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon↩
_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 4, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix39** = meos_mix_data(ident1 = "CO2", ident2 = "ACETONE", bibref = "", Fij = 1.6093, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩
11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩
0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix40** = meos_mix_data(ident1 = "CO2", ident2 = "AR", bibref = "", Fij = 1.0000000, num_mix = 6, n_mix = (/ -0.656000000000D-01,0.237000000000D-01,0.↩
352170000000D+01, -0.283100000000D+01,-0.140600000000D+01,0.864000000000D+00, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 3.2200000000,2.9000000000,1.9000000000, 1.5700000000,2.↩
7300000000,1.0800000000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 2,3,1,1,1,2, 0,0,0,0,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,1.2430000000, 1.0720000000,1.↩
4650000000,0.9460000000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.↩
5000000000, 0.5000000000,0.5000000000,0.5000000000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.6500000000, 0.7270000000,0.6480000000,0.7060000000, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,1.2080000000, 0.8200000000,1.5270000000,0.↩
8600000000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 4)

- type(meos_mix_data), parameter **meos_mix41** = meos_mix_data(ident1 = "N2", ident2 = "CO2", bibref = "", Fij = 1., num_mix = 6, n_mix = (/ 0.28661625028399,-0.10919833861247,-0.11374032082270d1, 0.↩ 76580544237358,0.42638000926819d-2,0.17673538204534, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_↩ mix = (/ 1.85,1.4,3.2, 2.5,8.0,3.75, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 2,3,1,1,1,2, 0,0,0,0,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.25, 0.25,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.5, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.5, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.↩ 0d0,0.0d0,0.75, 1.0d0,2.0d0,3.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 4)
- type(meos_mix_data), parameter **meos_mix42** = meos_mix_data(ident1 = "C1", ident2 = "BENZENE", bibref = "", Fij = 0.955, num_mix = 9, n_mix = (/ 0.013746429958576,-0.0074425012129552,-0.↩ 0045516600213685, -0.0054546603350237,0.0023682016824471,0.18007763721438, -0.44773942932486,0.↩ 019327374888200,-0.30632197804624, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 1.85,3.95,0.0, 1.85,3.85,5.25, 3.↩ 85,0.2,6.5, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 3,3,4,4,4,1, 1,1,2,0,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.25, 0.25,0.0,0.0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.5, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.5, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.75, 1.0,2.0,3.0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 4)
- type(meos_mix_data), parameter **meos_mix43** = meos_mix_data(ident1 = "BENZENE", ident2 = "NC7", bibref = "", Fij = -0.675, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩ 0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix44** = meos_mix_data(ident1 = "N2", ident2 = "CYCLOHEX", bibref = "", Fij = 0.735, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩ 0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix45** = meos_mix_data(ident1 = "C2", ident2 = "ACETONE", bibref = "", Fij = 2.1862, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩ 0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix46** = meos_mix_data(ident1 = "C1", ident2 = "MEOH", bibref = "", Fij = 1.276, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩ 0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix47** = meos_mix_data(ident1 = "C1", ident2 = "H2O", bibref = "", Fij = 1.0000000, num_mix = 6, n_mix = (/ 0.330000000000D+01,0.960000000000D+01,-0.↩ 117000000000D+02, 0.213000000000D+01,-0.530000000000D+00,-0.288000000000D+01, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 1.1000000000,0.8000000000,1.0000000000, 4.0000000000,3.↩ 4000000000,0.8000000000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,4,1, 0,0,0,0,0,0 /),

l_mix = (/ 0,0,1,1,1,0, 0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 4, num_gauss = 0)

- type(meos_mix_data), parameter **meos_mix48** = meos_mix_data(ident1 = "KR", ident2 = "CO2", bibref = "", Fij = 0.6362, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix49** = meos_mix_data(ident1 = "C3", ident2 = "ETOH", bibref = "", Fij = 1.15, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix50** = meos_mix_data(ident1 = "CO2", ident2 = "R1234ZE", bibref = "", Fij = -0.084, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix51** = meos_mix_data(ident1 = "C1", ident2 = "C3", bibref = "", Fij = 1., num_mix = 9, n_mix = (/ 0.13746429958576d-1,-0.74425012129552d-2,-0.45516600213685d-2, -0.54546603350237d-2,0.23682016824471d-2,0.18007763721438, -0.44773942932486,0.19327374888200d-1,-0.30632197804624, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 1.85,3.95,0.0, 1.85,3.85,5.25, 3.85,0.2,6.5, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 3,3,4,4,4,1, 1,1,2,0,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.25, 0.25,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.5, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.5, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.75, 1.0d0,2.0d0,3.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 4)
- type(meos_mix_data), parameter **meos_mix52** = meos_mix_data(ident1 = "C3", ident2 = "MEOH", bibref = "", Fij = 2.715, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix53** = meos_mix_data(ident1 = "C1", ident2 = "TOLU", bibref = "", Fij = 1.313, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0,

0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)

- type(meos_mix_data), parameter **meos_mix54** = meos_mix_data(ident1 = "CYCLOHEX", ident2 = "TOLU", bibref = "", Fij = 0.5865, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩ 0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix55** = meos_mix_data(ident1 = "H2S", ident2 = "NC12", bibref = "", Fij = -1.808, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩ 0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix56** = meos_mix_data(ident1 = "TOLU", ident2 = "NC10", bibref = "", Fij = -1.395, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩ 0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix57** = meos_mix_data(ident1 = "CYCLOHEX", ident2 = "NC7", bibref = "", Fij = -0.2539, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩ 0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix58** = meos_mix_data(ident1 = "IC5", ident2 = "MEOH", bibref = "", Fij = 43.215000, num_mix = 4, n_mix = (/ -0.013073,0.018259,0.81299e-5, 0.0078496,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 7.4,0.35,10.0, 5.3,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0 /), d_mix = (/ 1,3,11,2,0,0, 0,0,0,0,0,0 /), l_mix = (/ 1,1,2,3,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 4, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix59** = meos_mix_data(ident1 = "MEOH", ident2 = "CYCLOHEX", bibref = "", Fij = 15.061, num_mix = 4, n_mix = (/ -0.013073,0.018259,0.81299e-5, 0.0078496,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 7.4,0.35,10.0, 5.3,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩ 0d0,0.0d0,0.0d0 /), d_mix = (/ 1,3,11,2,0,0, 0,0,0,0,0,0 /), l_mix = (/ 1,1,2,3,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 4, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix60** = meos_mix_data(ident1 = "O2", ident2 = "H2O", bibref = "", Fij = 0.6017000, num_mix = 5, n_mix = (/ 0.401420790000D+01,-0.115739390000D+01,-0.↩ 721024250000D+01, -0.532512230000D+01,-0.221558670000D+01,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0 /), t_mix = (/ 0.5470000000,0.0550000000,1.9250000000, 0.5520000000,1.0000000000,0.↩

0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,4,0, 0,0,0,0,0,0 /), l_mix = (/ 0,1,1,1,1,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon↩
_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 4, num_gauss = 0)

- type(meos_mix_data), parameter **meos_mix61** = meos_mix_data(ident1 = "MEOH", ident2 = "NC6", bibref = "", Fij = 11.089885, num_mix = 4, n_mix = (/ -0.013073,0.018259,0.81299e-5, 0.0078496,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 7.4,0.35,10.0, 5.3,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.↩
0d0,0.0d0,0.0d0 /), d_mix = (/ 1,3,11,2,0,0, 0,0,0,0,0,0 /), l_mix = (/ 1,1,2,3,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 4, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix62** = meos_mix_data(ident1 = "CYCLOHEX", ident2 = "NC8", bibref = "", Fij = -0.358, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩
11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩
0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix63** = meos_mix_data(ident1 = "NC6", ident2 = "BENZENE", bibref = "", Fij = -0.92, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩
11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩
0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix64** = meos_mix_data(ident1 = "R32", ident2 = "R134A", bibref = "", Fij = 1., num_mix = 4, n_mix = (/ 0.22909,0.094074,0.00039876, 0.021133,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 1.9,0.25,0.07, 2.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,3,8,1,0,0, 0,0,0,0,0,0 /), l_mix = (/ 1,1,1,2,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.↩
0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 4, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix65** = meos_mix_data(ident1 = "H2S", ident2 = "H2O", bibref = "", Fij = 1.000000, num_mix = 6, n_mix = (/ 0.170000000000D+00,-0.111600000000D+00,0.↩
121000000000D+00, -0.235200000000D-02,-0.431000000000D-01,0.776400000000D+00, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 0.9000000000,4.0400000000,6.8800000000, 8.1500000000,5.↩
3500000000,2.7000000000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,1,2,4,8,1, 0,0,0,0,0,0 /), l_mix = (/ 0,0,1,1,2,1, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 4, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix66** = meos_mix_data(ident1 = "ETOH", ident2 = "MEG", bibref = "", Fij = 0.3312, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩
11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩
0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0,

0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)

- type(meos_mix_data), parameter **meos_mix67** = meos_mix_data(ident1 = "NC6", ident2 = "TOLU", bibref = "", Fij = -0.213, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩
11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩
0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix68** = meos_mix_data(ident1 = "NC4", ident2 = "MEOH", bibref = "", Fij = 1.068, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩
11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩
0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix69** = meos_mix_data(ident1 = "NC6", ident2 = "CYCLOHEX", bibref = "", Fij = -0.3672, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩
11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩
0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix70** = meos_mix_data(ident1 = "C1", ident2 = "C2", bibref = "", Fij = 1., num_mix = 12, n_mix = (/ -0.80926050298746d-3,-0.75381925080059d-3,-0.41618768891219d-1, -0.↩
23452173681569,0.14003840584586,0.63281744807738d-1, -0.34660425848809d-1,-0.23918747334251,0.↩
19855255066891d-2, 0.61777746171555d1,-0.69575358271105d1,0.10630185306388d1 /), t_mix = (/ 0.↩
65,1.55,3.1, 5.9,7.05,3.35, 1.2,5.8,2.7, 0.45,0.55,1.95 /), d_mix = (/ 3,4,1,2,2,2, 2,2,2,3,3,3 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,1.0d0, 1.0d0,1.0d0,0.875, 0.75,0.5,0.0d0, 0.0d0,0.↩
0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.5, 0.5,0.5,0.5, 0.5,0.5,0.5, 0.5,0.5,0.5 /), epsilon_mix = (/ 0.0d0,0.0d0,0.5, 0.5,0.5,0.5, 0.5,0.5,0.5, 0.5,0.5,0.5 /), beta_mix = (/ 0.0d0,0.0d0,1.0d0, 1.0d0,1.0d0,1.25, 1.5,2.0d0,3.0d0, 3.0d0,3.0d0,3.0d0 /), num_exp = 0, num_gauss = 10)
- type(meos_mix_data), parameter **meos_mix71** = meos_mix_data(ident1 = "NC9", ident2 = "NC12", bibref = "", Fij = 0.15, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩
11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩
0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix72** = meos_mix_data(ident1 = "CO2", ident2 = "CO", bibref = "", Fij = 1.0000000, num_mix = 6, n_mix = (/ 0.186100000000D+01,-0.401700000000D+01,0.↩
273400000000D+00, 0.239300000000D+01,0.264600000000D+02,-0.121300000000D+01, 0.0d0,0.↩
0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 2.8200000000,3.2600000000,0.9400000000, 3.9440000000,2.↩
5300000000,4.3800000000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,1,2,4,1,1, 0,0,0,0,0,0 /), l_mix = (/ 0,0,1,1,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,-0.3850000000,-0.2950000000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.1090000000,2.↩
5960000000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,-0.↩
1440000000,-0.3100000000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,5.1000000000,1.6610000000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 2, num_gauss = 2)

- type(meos_mix_data), parameter **meos_mix73** = meos_mix_data(ident1 = "MEOH", ident2 = "NC10", bibref = "", Fij = -7.124, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩ 0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix74** = meos_mix_data(ident1 = "N2", ident2 = "H2O", bibref = "", Fij = 1.0000000, num_mix = 5, n_mix = (/ 0.401420790000D+01,-0.115739390000D+01,-0.↩ 721024250000D+01, -0.532512230000D+01,-0.221558670000D+01,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0 /), t_mix = (/ 0.5470000000,0.0550000000,1.9250000000, 0.5520000000,1.0000000000,0.↩ 0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,4,0, 0,0,0,0,0,0 /), l_mix = (/ 0,1,1,1,1,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon↩ _mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 4, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix75** = meos_mix_data(ident1 = "CO2", ident2 = "MEOH", bibref = "", Fij = 1.77, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩ 0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix76** = meos_mix_data(ident1 = "BENZENE", ident2 = "NC10", bibref = "", Fij = -2.444, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩ 0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix77** = meos_mix_data(ident1 = "ETOH", ident2 = "TOLU", bibref = "", Fij = 0.282, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩ 0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix78** = meos_mix_data(ident1 = "BENZENE", ident2 = "NC8", bibref = "", Fij = -0.924, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.↩ 11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.↩ 0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.↩ 0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix79** = meos_mix_data(ident1 = "BENZENE", ident2 = "CYCLOHEX", bibref = "", Fij = -0.6475, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.↩ 7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.↩

10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)

- type(meos_mix_data), parameter **meos_mix80** = meos_mix_data(ident1 = "NC6", ident2 = "NC12", bibref = "", Fij = 1.14, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix81** = meos_mix_data(ident1 = "CO2", ident2 = "NC12", bibref = "", Fij = -1.539, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix82** = meos_mix_data(ident1 = "MEOH", ident2 = "MEG", bibref = "", Fij = -1.703, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix83** = meos_mix_data(ident1 = "BENZENE", ident2 = "NC9", bibref = "", Fij = -1.193, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix84** = meos_mix_data(ident1 = "C3", ident2 = "IC4", bibref = "", Fij = -0.0551609771024, num_mix = 10, n_mix = (/ 0.25574776844118d1,-0.79846357136353d1,0.47859131465806d1, -0.73265392369587,0.13805471345312d1,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.55527385721943d-4,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix85** = meos_mix_data(ident1 = "R32", ident2 = "R1234ZE", bibref = "", Fij = -0.265419, num_mix = 9, n_mix = (/ 0.013746429958576,-0.0074425012129552,-0.0045516600213685, -0.0054546603350237,0.0023682016824471,0.18007763721438, -0.44773942932486,0.019327374888200,-0.30632197804624, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 1.85,3.95,0.0, 1.85,3.85,5.25, 3.85,0.2,6.5, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 3,3,4,4,4,1, 1,1,2,0,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0

/), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.25, 0.25,0.0,0.0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.5, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.5, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.75, 1.0,2.0,3.0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 4)

- type(meos_mix_data), parameter **meos_mix86** = meos_mix_data(ident1 = "C1", ident2 = "H2", bibref = "", Fij = 1.000000, num_mix = 8, n_mix = (/ 0.128000000000D+01,-0.774000000000D+00,-0.914000000000D+00, 0.360000000000D+00,-0.245000000000D+01,0.446200000000D+01, -0.972000000000D+00,-0.207000000000D+01,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 0.3400000000,0.4200000000,0.8000000000, 3.2900000000,0.0500000000,0.2540000000, 0.4100000000,2.6300000000,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,2,1,2,1,2, 3,1,0,0,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,-0.8300000000,-0.3400000000, -0.5700000000,-0.4400000000,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,1.3600000000,1.4400000000, 1.6900000000,1.5300000000,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,-0.9700000000,-0.2000000000, -0.2600000000,-0.1800000000,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.7700000000,1.7400000000, 0.7900000000,0.4000000000,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 4)
- type(meos_mix_data), parameter **meos_mix87** = meos_mix_data(ident1 = "ETOH", ident2 = "BENZENE", bibref = "", Fij = -0.162, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix88** = meos_mix_data(ident1 = "C1", ident2 = "CYCLOHEX", bibref = "", Fij = -0.3986, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix89** = meos_mix_data(ident1 = "HE", ident2 = "KR", bibref = "", Fij = 1.41, num_mix = 4, n_mix = (/ -0.25157134971934,-0.0062203841111983,0.088850315184396, -0.035592212573239,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 2.0,-1.0,1.75, 1.4,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,3,3,4,0,0, 0,0,0,0,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix90** = meos_mix_data(ident1 = "ETOH", ident2 = "H2O", bibref = "", Fij = 1.000000, num_mix = 6, n_mix = (/ -0.272600000000D+00,0.270000000000D-01,-0.148300000000D-01, 0.177300000000D+01,0.690000000000D+01,-0.642000000000D+01, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 1.6800000000,0.7300000000,4.5500000000, 1.1700000000,0.1500000000,0.4300000000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,4,3,2,1,1, 0,0,0,0,0,0 /), l_mix = (/ 0,0,1,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, -0.58500,-0.51000,-0.70000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 1.0800,0.7500,1.3400, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, -0.1900,-2.1200,-1.2200, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 1.11000,1.64000,1.64000, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 1, num_gauss = 3)
- type(meos_mix_data), parameter **meos_mix91** = meos_mix_data(ident1 = "C3", ident2 = "NC4", bibref = "", Fij = 0.0312572600489, num_mix = 10, n_mix = (/ 0.25574776844118d1,-0.79846357136353d1,0.47859131465806d1, -0.73265392369587,0.13805471345312d1,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.55527385721943d-4,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0,

0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)

- type(meos_mix_data), parameter **meos_mix92** = meos_mix_data(ident1 = "NE", ident2 = "N2", bibref = "", Fij = 1.7, num_mix = 6, n_mix = (/ -0.47376518126608,0.48961193461001,-0.0057011062090535, -0.19966820041320,-0.69411103101723,0.69226192739021, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 0.0,0.05,0.0, 3.65,4.9,4.45, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 2,2,3,1,2,2, 0,0,0,0,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 1.0,1.0,0.875, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 1.0,1.0,1.25, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 3)
- type(meos_mix_data), parameter **meos_mix93** = meos_mix_data(ident1 = "N2", ident2 = "H2", bibref = "", Fij = 1.000000, num_mix = 8, n_mix = (/ -0.165900000000D+01,-0.268000000000D+00,-0.502000000000D+00, 0.323000000000D+00,0.297600000000D+01,0.508400000000D+01, -0.386600000000D+01,-0.547400000000D+01,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0 /), t_mix = (/ 0.7240000000,0.0970000000,2.9530000000, 3.5000000000,2.9390000000,0.6940000000, 2.3290000000,1.0660000000,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,2,1,2,1,1, 1,1,0,0,0,0 /), l_mix = (/ 1,1,2,2,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,-1.8600000000,-0.0300000000, -1.8500000000,-0.1300000000,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,2.5100000000,0.3900000000, 2.5100000000,1.2200000000,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,-0.9300000000,-0.3600000000, -0.9900000000,-0.2400000000,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,1.3000000000,0.8000000000, 1.5000000000,0.2300000000,0.0d0, 0.0d0,0.0d0,0.0d0,0.0d0 /), num_exp = 4, num_gauss = 4)
- type(meos_mix_data), parameter **meos_mix94** = meos_mix_data(ident1 = "C1", ident2 = "IC4", bibref = "", Fij = 0.771035405688, num_mix = 10, n_mix = (/ 0.25574776844118d1,-0.79846357136353d1,0.47859131465806d1, -0.73265392369587,0.13805471345312d1,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.55527385721943d-4,0.0d0,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix95** = meos_mix_data(ident1 = "NH3", ident2 = "H2O", bibref = "", Fij = 1., num_mix = 7, n_mix = (/ -2.00211,3.08130,-1.75352, 2.98160,-3.82588,-1.73850, 0.42008,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 0.25,2.0,0.5, 2.0,1.0,4.0, 1.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,1,1,1, 3,0,0,0,0,0 /), l_mix = (/ 2,1,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0, -0.746,-4.25,-0.7, -3.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0, 2.0,-0.25,1.85, 0.3,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,-0.27, -0.86,-3.0,-0.5, -4.0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,2.8, 1.8,1.5,0.8, 1.3,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 2, num_gauss = 5)
- type(meos_mix_data), parameter **meos_mix96** = meos_mix_data(ident1 = "H2", ident2 = "NE", bibref = "", Fij = 0.1617, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix97** = meos_mix_data(ident1 = "C1", ident2 = "CO2", bibref = "", Fij = 1., num_mix = 6, n_mix = (/ -0.10859387354942,0.80228576727389d-1,-0.93303985115717d-2, 0.40989274005848d-1,-0.24338019772494,0.23855347281124, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 2.6,1.95,0.0, 3.95,7.95,8.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,2,3,1,2,3, 0,0,0,0,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 1.0d0,0.5,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0

/), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 1.0d0,2.0d0,3.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 3)

- type(meos_mix_data), parameter **meos_mix98** = meos_mix_data(ident1 = "R32", ident2 = "R1234YF", bibref = "", Fij = -0.277708, num_mix = 6, n_mix = (/ -0.10859387354942,0.080228576727389,-0.0093303985115717, 0.040989274005848,-0.24338019772494,0.23855347281124, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 2.6,1.95,0.0, 3.95,7.95,8.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,2,3,1,2,3, 0,0,0,0,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 1.0,0.5,0.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.5,0.5,0.5, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 1.0,2.0,3.0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 3)
- type(meos_mix_data), parameter **meos_mix99** = meos_mix_data(ident1 = "CO2", ident2 = "CYCLOHEX", bibref = "", Fij = 0.0184, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix100** = meos_mix_data(ident1 = "CO2", ident2 = "H2O", bibref = "", Fij = 1.0000000, num_mix = 8, n_mix = (/ 0.394404670000D+00,-0.176347320000D+01,0.146207550000D+00, 0.875223200000D-02,0.203493980000D+01,-0.903502500000D-01, -0.216388540000D+00,0.396121700000D-01,0.0d0, 0.0d0,0.0d0,0.0d0 /), t_mix = (/ 0.8800000000,2.9320000000,2.4330000000, 1.3300000000,4.4160000000,5.5140000000, 5.2030000000,1.0000000000,0.0d0, 0.0d0,0.0d0,0.0d0 /), d_mix = (/ 1,1,3,0,2,3, 1,5,0,0,0,0 /), l_mix = (/ 0,0,0,1,1,1, 2,2,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 5, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix101** = meos_mix_data(ident1 = "C2", ident2 = "MEOH", bibref = "", Fij = 2.751, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix102** = meos_mix_data(ident1 = "MEOH", ident2 = "TOLU", bibref = "", Fij = 5.902, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- type(meos_mix_data), parameter **meos_mix103** = meos_mix_data(ident1 = "CYCLOHEX", ident2 = "NC10", bibref = "", Fij = -0.358, num_mix = 10, n_mix = (/ 2.5574776844118,-7.9846357136353,4.7859131465806, -0.73265392369587,1.3805471345312,0.28349603476365, -0.49087385940425,-0.10291888921447,0.11836314681968, 0.000055527385721943,0.0d0,0.0d0 /), t_mix = (/ 1.0,1.55,1.7, 0.25,1.35,0.0, 1.25,0.0,0.7, 5.4,0.0d0,0.0d0 /), d_mix = (/ 1,1,1,2,2,3, 3,4,4,4,0,0 /), l_mix = (/ 0,0,0,0,0,0, 0,0,0,0,0,0 /), eta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), gamma_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), epsilon_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), beta_mix = (/ 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0, 0.0d0,0.0d0,0.0d0 /), num_exp = 0, num_gauss = 0)
- integer, parameter **max_meos_mix_data** = 103

- type(meos_mix_data), dimension(max_meos_mix_data), parameter **meos_mix_datadb** = (/ meos↩
_mix1,meos_mix2,meos_mix3,meos_mix4,meos_mix5,meos_mix6, meos_mix7,meos_mix8,meos_↩
mix9,meos_mix10,meos_mix11,meos_mix12, meos_mix13,meos_mix14,meos_mix15,meos_mix16,meos↩
_mix17,meos_mix18, meos_mix19,meos_mix20,meos_mix21,meos_mix22,meos_mix23,meos_mix24,
meos_mix25,meos_mix26,meos_mix27,meos_mix28,meos_mix29,meos_mix30, meos_mix31,meos_↩
mix32,meos_mix33,meos_mix34,meos_mix35,meos_mix36, meos_mix37,meos_mix38,meos_mix39,meos↩
_mix40,meos_mix41,meos_mix42, meos_mix43,meos_mix44,meos_mix45,meos_mix46,meos_mix47,meos↩
_mix48, meos_mix49,meos_mix50,meos_mix51,meos_mix52,meos_mix53,meos_mix54, meos_↩
mix55,meos_mix56,meos_mix57,meos_mix58,meos_mix59,meos_mix60, meos_mix61,meos_mix62,meos↩
_mix63,meos_mix64,meos_mix65,meos_mix66, meos_mix67,meos_mix68,meos_mix69,meos_mix70,meos↩
_mix71,meos_mix72, meos_mix73,meos_mix74,meos_mix75,meos_mix76,meos_mix77,meos_mix78,
meos_mix79,meos_mix80,meos_mix81,meos_mix82,meos_mix83,meos_mix84, meos_mix85,meos_↩
mix86,meos_mix87,meos_mix88,meos_mix89,meos_mix90, meos_mix91,meos_mix92,meos_mix93,meos↩
_mix94,meos_mix95,meos_mix96, meos_mix97,meos_mix98,meos_mix99,meos_mix100,meos_mix101,meos↩
_mix102, meos_mix103 /)

5.33.1 Detailed Description

Automatically generated file meosmixdb.f90 Time stamp: 2023-03-23T14:50:21.734496.

5.34 mixdatadb Module Reference

Automatically generated to file mixdatadb.f90 using utility python code pyUtils Time stamp: 2023-02-27T15:11↩
:46.339247.

Variables

- type(kijdatadb), parameter **vdw1** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "H2S", kijvalue = 0.09500000)
- type(kijdatadb), parameter **vdw2** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "O2", kijvalue = 0.10900000)
- type(kijdatadb), parameter **vdw3** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NH3", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw4** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "N2O", kijvalue = 0.01000000)
- type(kijdatadb), parameter **vdw5** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "N2O4", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw6** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "C1", kijvalue = 0.10000000)
- type(kijdatadb), parameter **vdw7** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "C2", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw8** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "C2_1", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw9** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "C3", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw10** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "H2O", kijvalue = 0.07400000)
- type(kijdatadb), parameter **vdw11** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "IC4", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw12** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "IC5", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw13** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC10", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw14** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC11", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw15** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC4", kijvalue = 0.15000000)

- type(kijdatadb), parameter **vdw16** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC5", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw17** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "H2S", kijvalue = 0.10200000)
- type(kijdatadb), parameter **vdw18** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "N2", kijvalue = -0.03600000)
- type(kijdatadb), parameter **vdw19** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "AR", kijvalue = 0.08800000)
- type(kijdatadb), parameter **vdw20** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "SO2", kijvalue = 0.07500000)
- type(kijdatadb), parameter **vdw21** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "H2", kijvalue = 0.12100000)
- type(kijdatadb), parameter **vdw22** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "CO", kijvalue = -0.05900000)
- type(kijdatadb), parameter **vdw23** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NH3", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw24** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC4", kijvalue = 0.06000000)
- type(kijdatadb), parameter **vdw25** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC4", kijvalue = 0.06000000)
- type(kijdatadb), parameter **vdw26** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC5", kijvalue = 0.06000000)
- type(kijdatadb), parameter **vdw27** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC6", kijvalue = 0.05000000)
- type(kijdatadb), parameter **vdw28** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC7", kijvalue = 0.04000000)
- type(kijdatadb), parameter **vdw29** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC8", kijvalue = 0.04000000)
- type(kijdatadb), parameter **vdw30** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC9", kijvalue = 0.03000000)
- type(kijdatadb), parameter **vdw31** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MEG", uid2 = "CO2", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw32** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MEG", uid2 = "H2O", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw33** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MEG", uid2 = "C1", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw34** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MEOH", uid2 = "CO2", kijvalue = 0.01700000)
- type(kijdatadb), parameter **vdw35** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "C1", kijvalue = 0.03800000)
- type(kijdatadb), parameter **vdw36** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "C2", kijvalue = 0.04100000)
- type(kijdatadb), parameter **vdw37** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "C2_1", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw38** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "C3", kijvalue = 0.07600000)
- type(kijdatadb), parameter **vdw39** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "IC4", kijvalue = 0.09400000)
- type(kijdatadb), parameter **vdw40** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "IC5", kijvalue = 0.08700000)
- type(kijdatadb), parameter **vdw41** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC10", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw42** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC11", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw43** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC4", kijvalue = 0.07000000)

- type(kijdatadb), parameter **vdw44** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC5", kijvalue = 0.08800000)
- type(kijdatadb), parameter **vdw45** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC6", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw46** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC7", kijvalue = 0.14200000)
- type(kijdatadb), parameter **vdw47** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC8", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw48** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC9", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw49** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "O2", kijvalue = -0.01300000)
- type(kijdatadb), parameter **vdw50** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "BENZENE", kijvalue = 0.01100000)
- type(kijdatadb), parameter **vdw51** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R12", uid2 = "R11", kijvalue = 0.00540000)
- type(kijdatadb), parameter **vdw52** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R12", uid2 = "R114", kijvalue = 0.00150000)
- type(kijdatadb), parameter **vdw53** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R12", uid2 = "R152a", kijvalue = 0.08670000)
- type(kijdatadb), parameter **vdw54** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R13", uid2 = "R11", kijvalue = 0.02620000)
- type(kijdatadb), parameter **vdw55** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R13", uid2 = "R113", kijvalue = 0.02430000)
- type(kijdatadb), parameter **vdw56** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R13", uid2 = "R12", kijvalue = 0.02990000)
- type(kijdatadb), parameter **vdw57** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R13B", uid2 = "R12", kijvalue = -0.00320000)
- type(kijdatadb), parameter **vdw58** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R13B", uid2 = "R152a", kijvalue = 0.07990000)
- type(kijdatadb), parameter **vdw59** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R14", uid2 = "R13", kijvalue = 0.03040000)
- type(kijdatadb), parameter **vdw60** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R14", uid2 = "R23", kijvalue = 0.10080000)
- type(kijdatadb), parameter **vdw61** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R11", kijvalue = 0.04660000)
- type(kijdatadb), parameter **vdw62** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R114", kijvalue = 0.03990000)
- type(kijdatadb), parameter **vdw63** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R12", kijvalue = 0.05640000)
- type(kijdatadb), parameter **vdw64** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R142b", kijvalue = 0.00570000)
- type(kijdatadb), parameter **vdw65** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R115", kijvalue = 0.08700000)
- type(kijdatadb), parameter **vdw66** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R23", uid2 = "R13", kijvalue = 0.10320000)
- type(kijdatadb), parameter **vdw67** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R218", uid2 = "R152a", kijvalue = 0.12000000)
- type(kijdatadb), parameter **vdw68** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R125", uid2 = "R143a", kijvalue = -0.01110000)
- type(kijdatadb), parameter **vdw69** = kijdatadb(eosid = "CSP-PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R125", uid2 = "R134a", kijvalue = -0.00240000)
- type(kijdatadb), parameter **vdw70** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "H2S", kijvalue = 0.10100000)
- type(kijdatadb), parameter **vdw71** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "C2", kijvalue = -0.00780000)

- type(kijdatadb), parameter **vdw72** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "C2_1", kijvalue = 0.02000000)
- type(kijdatadb), parameter **vdw73** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "C3", kijvalue = 0.00900000)
- type(kijdatadb), parameter **vdw74** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "IC4", kijvalue = 0.02410000)
- type(kijdatadb), parameter **vdw75** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "NC4", kijvalue = 0.00560000)
- type(kijdatadb), parameter **vdw76** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "NC5", kijvalue = 0.01900000)
- type(kijdatadb), parameter **vdw77** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "H2S", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw78** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "C2_1", kijvalue = 0.01120000)
- type(kijdatadb), parameter **vdw79** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "C3", kijvalue = -0.00220000)
- type(kijdatadb), parameter **vdw80** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "IC4", kijvalue = -0.01000000)
- type(kijdatadb), parameter **vdw81** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "NC4", kijvalue = 0.00670000)
- type(kijdatadb), parameter **vdw82** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "NC5", kijvalue = 0.00560000)
- type(kijdatadb), parameter **vdw83** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "C2_1", kijvalue = 0.10000000)
- type(kijdatadb), parameter **vdw84** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "IC4", kijvalue = -0.01000000)
- type(kijdatadb), parameter **vdw85** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "NC5", kijvalue = 0.02300000)
- type(kijdatadb), parameter **vdw86** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "NC4", kijvalue = 0.00110000)
- type(kijdatadb), parameter **vdw87** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "NC5", kijvalue = 0.02040000)
- type(kijdatadb), parameter **vdw88** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2O", uid2 = "C1", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw89** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "O2", kijvalue = 0.11800000)
- type(kijdatadb), parameter **vdw90** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NH3", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw91** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "N2O", kijvalue = 0.01000000)
- type(kijdatadb), parameter **vdw92** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "N2O4", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw93** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "CO", kijvalue = -0.06800000)
- type(kijdatadb), parameter **vdw94** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "C1", kijvalue = 0.10600000)
- type(kijdatadb), parameter **vdw95** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "C2", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw96** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "C2_1", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw97** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "C3", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw98** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "H2O", kijvalue = 0.07400000)
- type(kijdatadb), parameter **vdw99** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "IC4", kijvalue = 0.15000000)

- type(kijdatadb), parameter **vdw100** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "IC5", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw101** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC10", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw102** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC11", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw103** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC4", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw104** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC5", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw105** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "H2S", kijvalue = 0.09900000)
- type(kijdatadb), parameter **vdw106** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "N2", kijvalue = -0.04200000)
- type(kijdatadb), parameter **vdw107** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "AR", kijvalue = 0.09400000)
- type(kijdatadb), parameter **vdw108** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "SO2", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw109** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "H2", kijvalue = 0.10800000)
- type(kijdatadb), parameter **vdw110** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC4", kijvalue = 0.06000000)
- type(kijdatadb), parameter **vdw111** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC5", kijvalue = 0.06000000)
- type(kijdatadb), parameter **vdw112** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC6", kijvalue = 0.05000000)
- type(kijdatadb), parameter **vdw113** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC7", kijvalue = 0.04000000)
- type(kijdatadb), parameter **vdw114** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC8", kijvalue = 0.04000000)
- type(kijdatadb), parameter **vdw115** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC9", kijvalue = 0.03000000)
- type(kijdatadb), parameter **vdw116** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MEG", uid2 = "CO2", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw117** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MEG", uid2 = "H2O", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw118** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MEG", uid2 = "C1", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw119** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MEOH", uid2 = "CO2", kijvalue = 0.01700000)
- type(kijdatadb), parameter **vdw120** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "C1", kijvalue = 0.04200000)
- type(kijdatadb), parameter **vdw121** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "C2", kijvalue = 0.06000000)
- type(kijdatadb), parameter **vdw122** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "C2_1", kijvalue = 0.07500000)
- type(kijdatadb), parameter **vdw123** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "C3", kijvalue = 0.09000000)
- type(kijdatadb), parameter **vdw124** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "IC4", kijvalue = 0.11300000)
- type(kijdatadb), parameter **vdw125** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "IC5", kijvalue = 0.08700000)
- type(kijdatadb), parameter **vdw126** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC10", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw127** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC11", kijvalue = 0.08000000)

- type(kijdatadb), parameter **vdw128** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC4", kijvalue = 0.11300000)
- type(kijdatadb), parameter **vdw129** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC5", kijvalue = 0.14000000)
- type(kijdatadb), parameter **vdw130** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC6", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw131** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC7", kijvalue = 0.14200000)
- type(kijdatadb), parameter **vdw132** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC8", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw133** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC9", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw134** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "O2", kijvalue = -0.00800000)
- type(kijdatadb), parameter **vdw135** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "BENZENE", kijvalue = 0.01100000)
- type(kijdatadb), parameter **vdw136** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R12", uid2 = "R11", kijvalue = 0.00540000)
- type(kijdatadb), parameter **vdw137** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R12", uid2 = "R114", kijvalue = 0.00150000)
- type(kijdatadb), parameter **vdw138** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R12", uid2 = "R152a", kijvalue = 0.08670000)
- type(kijdatadb), parameter **vdw139** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R13", uid2 = "R11", kijvalue = 0.02620000)
- type(kijdatadb), parameter **vdw140** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R13", uid2 = "R113", kijvalue = 0.02430000)
- type(kijdatadb), parameter **vdw141** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R13", uid2 = "R12", kijvalue = 0.02990000)
- type(kijdatadb), parameter **vdw142** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R13B", uid2 = "R12", kijvalue = -0.00320000)
- type(kijdatadb), parameter **vdw143** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R13B", uid2 = "R152a", kijvalue = 0.07990000)
- type(kijdatadb), parameter **vdw144** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R14", uid2 = "R13", kijvalue = 0.03040000)
- type(kijdatadb), parameter **vdw145** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R14", uid2 = "R23", kijvalue = 0.10080000)
- type(kijdatadb), parameter **vdw146** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R11", kijvalue = 0.04660000)
- type(kijdatadb), parameter **vdw147** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R114", kijvalue = 0.03990000)
- type(kijdatadb), parameter **vdw148** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R12", kijvalue = 0.05640000)
- type(kijdatadb), parameter **vdw149** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R142b", kijvalue = 0.00570000)
- type(kijdatadb), parameter **vdw150** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R115", kijvalue = 0.08900000)
- type(kijdatadb), parameter **vdw151** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R23", uid2 = "R13", kijvalue = 0.10320000)
- type(kijdatadb), parameter **vdw152** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R218", uid2 = "R152a", kijvalue = 0.12000000)
- type(kijdatadb), parameter **vdw153** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R125", uid2 = "R143a", kijvalue = -0.01110000)
- type(kijdatadb), parameter **vdw154** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R125", uid2 = "R134a", kijvalue = -0.00240000)
- type(kijdatadb), parameter **vdw155** = kijdatadb(eosid = "CSP-SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R143a", uid2 = "R134a", kijvalue = 0.00130000)

- type(kijdatadb), parameter **vdw156** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C1", uid2 = "C2", kijvalue = 1.05200000)
- type(kijdatadb), parameter **vdw157** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C1", uid2 = "C2_1", kijvalue = 1.01400000)
- type(kijdatadb), parameter **vdw158** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C1", uid2 = "C3", kijvalue = 1.11300000)
- type(kijdatadb), parameter **vdw159** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C1", uid2 = "PRLN", kijvalue = 1.08900000)
- type(kijdatadb), parameter **vdw160** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C1", uid2 = "IC4", kijvalue = 1.15500000)
- type(kijdatadb), parameter **vdw161** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C1", uid2 = "NC4", kijvalue = 1.17100000)
- type(kijdatadb), parameter **vdw162** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C1", uid2 = "NC5", kijvalue = 1.24000000)
- type(kijdatadb), parameter **vdw163** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C1", uid2 = "IC5", kijvalue = 1.22800000)
- type(kijdatadb), parameter **vdw164** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C1", uid2 = "NC6", kijvalue = 1.30400000)
- type(kijdatadb), parameter **vdw165** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C1", uid2 = "NC7", kijvalue = 1.36700000)
- type(kijdatadb), parameter **vdw166** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C1", uid2 = "NC8", kijvalue = 1.42300000)
- type(kijdatadb), parameter **vdw167** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C1", uid2 = "NC9", kijvalue = 1.48400000)
- type(kijdatadb), parameter **vdw168** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C1", uid2 = "NC10", kijvalue = 1.53300000)
- type(kijdatadb), parameter **vdw169** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C1", uid2 = "BENZENE", kijvalue = 1.23400000)
- type(kijdatadb), parameter **vdw170** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C2", uid2 = "C2_1", kijvalue = 0.99100000)
- type(kijdatadb), parameter **vdw171** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C2", uid2 = "C3", kijvalue = 1.01000000)
- type(kijdatadb), parameter **vdw172** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C2", uid2 = "PRLN", kijvalue = 1.00200000)
- type(kijdatadb), parameter **vdw173** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C2", uid2 = "IC4", kijvalue = 1.03600000)
- type(kijdatadb), parameter **vdw174** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C2", uid2 = "NC4", kijvalue = 1.02900000)
- type(kijdatadb), parameter **vdw175** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C2", uid2 = "NC5", kijvalue = 1.06400000)
- type(kijdatadb), parameter **vdw176** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C2", uid2 = "IC5", kijvalue = 1.07000000)
- type(kijdatadb), parameter **vdw177** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C2", uid2 = "NC6", kijvalue = 1.10600000)
- type(kijdatadb), parameter **vdw178** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C2", uid2 = "NC7", kijvalue = 1.14300000)
- type(kijdatadb), parameter **vdw179** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C2", uid2 = "NC8", kijvalue = 1.16500000)
- type(kijdatadb), parameter **vdw180** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C2", uid2 = "NC9", kijvalue = 1.21400000)
- type(kijdatadb), parameter **vdw181** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C2", uid2 = "NC10", kijvalue = 1.23700000)
- type(kijdatadb), parameter **vdw182** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C2", uid2 = "BENZENE", kijvalue = 1.06600000)
- type(kijdatadb), parameter **vdw183** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C2_1", uid2 = "NC4", kijvalue = 0.99800000)

- type(kijdatadb), parameter **vdw184** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C2_1", uid2 = "BENZENE", kijvalue = 1.09400000)
- type(kijdatadb), parameter **vdw185** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C2_1", uid2 = "NC7", kijvalue = 1.16300000)
- type(kijdatadb), parameter **vdw186** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C3", uid2 = "PRLN", kijvalue = 0.99200000)
- type(kijdatadb), parameter **vdw187** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C3", uid2 = "NC4", kijvalue = 1.00300000)
- type(kijdatadb), parameter **vdw188** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C3", uid2 = "IC4", kijvalue = 1.00300000)
- type(kijdatadb), parameter **vdw189** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C3", uid2 = "NC5", kijvalue = 1.00600000)
- type(kijdatadb), parameter **vdw190** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C3", uid2 = "IC5", kijvalue = 1.00900000)
- type(kijdatadb), parameter **vdw191** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C3", uid2 = "NC6", kijvalue = 1.04700000)
- type(kijdatadb), parameter **vdw192** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C3", uid2 = "BENZENE", kijvalue = 1.01100000)
- type(kijdatadb), parameter **vdw193** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C3", uid2 = "NC7", kijvalue = 1.06700000)
- type(kijdatadb), parameter **vdw194** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C3", uid2 = "NC8", kijvalue = 1.09000000)
- type(kijdatadb), parameter **vdw195** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C3", uid2 = "NC9", kijvalue = 1.01150000)
- type(kijdatadb), parameter **vdw196** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "C3", uid2 = "NC10", kijvalue = 1.13900000)
- type(kijdatadb), parameter **vdw197** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "PRLN", uid2 = "NC4", kijvalue = 1.01000000)
- type(kijdatadb), parameter **vdw198** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "PRLN", uid2 = "IC4", kijvalue = 1.00900000)
- type(kijdatadb), parameter **vdw199** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC4", uid2 = "IC4", kijvalue = 1.00100000)
- type(kijdatadb), parameter **vdw200** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC4", uid2 = "NC5", kijvalue = 0.99400000)
- type(kijdatadb), parameter **vdw201** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC4", uid2 = "IC5", kijvalue = 0.99800000)
- type(kijdatadb), parameter **vdw202** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC4", uid2 = "NC6", kijvalue = 1.01800000)
- type(kijdatadb), parameter **vdw203** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC4", uid2 = "BENZENE", kijvalue = 0.99900000)
- type(kijdatadb), parameter **vdw204** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC4", uid2 = "NC7", kijvalue = 1.02700000)
- type(kijdatadb), parameter **vdw205** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC4", uid2 = "NC8", kijvalue = 1.04600000)
- type(kijdatadb), parameter **vdw206** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC4", uid2 = "NC9", kijvalue = 1.06400000)
- type(kijdatadb), parameter **vdw207** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC4", uid2 = "NC10", kijvalue = 1.07800000)
- type(kijdatadb), parameter **vdw208** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC5", uid2 = "IC5", kijvalue = 0.98700000)
- type(kijdatadb), parameter **vdw209** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC5", uid2 = "NC6", kijvalue = 0.99600000)
- type(kijdatadb), parameter **vdw210** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC5", uid2 = "BENZENE", kijvalue = 0.97700000)
- type(kijdatadb), parameter **vdw211** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC5", uid2 = "NC7", kijvalue = 1.00400000)

- type(kijdatadb), parameter **vdw212** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC5", uid2 = "NC8", kijvalue = 1.02000000)
- type(kijdatadb), parameter **vdw213** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC5", uid2 = "NC9", kijvalue = 1.03300000)
- type(kijdatadb), parameter **vdw214** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC5", uid2 = "NC10", kijvalue = 1.04500000)
- type(kijdatadb), parameter **vdw215** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC6", uid2 = "BENZENE", kijvalue = 0.97800000)
- type(kijdatadb), parameter **vdw216** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC6", uid2 = "NC7", kijvalue = 1.00800000)
- type(kijdatadb), parameter **vdw217** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC6", uid2 = "NC8", kijvalue = 1.00500000)
- type(kijdatadb), parameter **vdw218** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC6", uid2 = "NC9", kijvalue = 1.01500000)
- type(kijdatadb), parameter **vdw219** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC6", uid2 = "NC10", kijvalue = 1.02500000)
- type(kijdatadb), parameter **vdw220** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "BENZENE", uid2 = "NC7", kijvalue = 0.98500000)
- type(kijdatadb), parameter **vdw221** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "BENZENE", uid2 = "NC8", kijvalue = 0.98700000)
- type(kijdatadb), parameter **vdw222** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "BENZENE", uid2 = "NC9", kijvalue = 1.03400000)
- type(kijdatadb), parameter **vdw223** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "BENZENE", uid2 = "NC10", kijvalue = 1.04700000)
- type(kijdatadb), parameter **vdw224** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC7", uid2 = "NC8", kijvalue = 0.99300000)
- type(kijdatadb), parameter **vdw225** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC7", uid2 = "NC9", kijvalue = 1.00200000)
- type(kijdatadb), parameter **vdw226** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC7", uid2 = "NC10", kijvalue = 1.01000000)
- type(kijdatadb), parameter **vdw227** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC8", uid2 = "NC9", kijvalue = 0.99300000)
- type(kijdatadb), parameter **vdw228** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC8", uid2 = "NC10", kijvalue = 0.99900000)
- type(kijdatadb), parameter **vdw229** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "NC9", uid2 = "NC10", kijvalue = 0.99100000)
- type(kijdatadb), parameter **vdw230** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "N2", uid2 = "C1", kijvalue = 0.97700000)
- type(kijdatadb), parameter **vdw231** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "N2", uid2 = "C2_1", kijvalue = 1.03200000)
- type(kijdatadb), parameter **vdw232** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "N2", uid2 = "C2", kijvalue = 1.08200000)
- type(kijdatadb), parameter **vdw233** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "N2", uid2 = "C3", kijvalue = 1.17700000)
- type(kijdatadb), parameter **vdw234** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "N2", uid2 = "PRLN", kijvalue = 1.15100000)
- type(kijdatadb), parameter **vdw235** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "N2", uid2 = "NC4", kijvalue = 1.27600000)
- type(kijdatadb), parameter **vdw236** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "N2", uid2 = "NC5", kijvalue = 1.37200000)
- type(kijdatadb), parameter **vdw237** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "N2", uid2 = "NC6", kijvalue = 1.44200000)
- type(kijdatadb), parameter **vdw238** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "N2", uid2 = "O2", kijvalue = 0.99700000)
- type(kijdatadb), parameter **vdw239** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "N2", uid2 = "CO", kijvalue = 0.98700000)

- type(kijdatadb), parameter **vdw240** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "N2", uid2 = "H2S", kijvalue = 0.98300000)
- type(kijdatadb), parameter **vdw241** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "N2", uid2 = "CO2", kijvalue = 1.11000000)
- type(kijdatadb), parameter **vdw242** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "N2", uid2 = "N2O", kijvalue = 1.07300000)
- type(kijdatadb), parameter **vdw243** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "N2", uid2 = "NH3", kijvalue = 1.03300000)
- type(kijdatadb), parameter **vdw244** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "CO2", uid2 = "C1", kijvalue = 0.97500000)
- type(kijdatadb), parameter **vdw245** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "CO2", uid2 = "C2", kijvalue = 0.93800000)
- type(kijdatadb), parameter **vdw246** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "CO2", uid2 = "C3", kijvalue = 0.92500000)
- type(kijdatadb), parameter **vdw247** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "CO2", uid2 = "NC4", kijvalue = 0.95500000)
- type(kijdatadb), parameter **vdw248** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "CO2", uid2 = "IC4", kijvalue = 0.94600000)
- type(kijdatadb), parameter **vdw249** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "CO2", uid2 = "NC5", kijvalue = 1.00200000)
- type(kijdatadb), parameter **vdw250** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "CO2", uid2 = "NC6", kijvalue = 1.01800000)
- type(kijdatadb), parameter **vdw251** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "CO2", uid2 = "BENZENE", kijvalue = 1.01800000)
- type(kijdatadb), parameter **vdw252** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "CO2", uid2 = "NC7", kijvalue = 1.05800000)
- type(kijdatadb), parameter **vdw253** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "CO2", uid2 = "NC8", kijvalue = 1.09000000)
- type(kijdatadb), parameter **vdw254** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "CO2", uid2 = "NC9", kijvalue = 1.12600000)
- type(kijdatadb), parameter **vdw255** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "CO2", uid2 = "NC10", kijvalue = 1.16000000)
- type(kijdatadb), parameter **vdw256** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "CO2", uid2 = "H2S", kijvalue = 0.92200000)
- type(kijdatadb), parameter **vdw257** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "CO2", uid2 = "R12", kijvalue = 0.96900000)
- type(kijdatadb), parameter **vdw258** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "CO2", uid2 = "MEOH", kijvalue = 1.06900000)
- type(kijdatadb), parameter **vdw259** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "H2", uid2 = "C1", kijvalue = 1.21600000)
- type(kijdatadb), parameter **vdw260** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "H2", uid2 = "C2", kijvalue = 1.60400000)
- type(kijdatadb), parameter **vdw261** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "H2", uid2 = "PRLN", kijvalue = 1.49800000)
- type(kijdatadb), parameter **vdw262** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "H2", uid2 = "C3", kijvalue = 1.82600000)
- type(kijdatadb), parameter **vdw263** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "H2", uid2 = "NC4", kijvalue = 2.09300000)
- type(kijdatadb), parameter **vdw264** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "H2", uid2 = "NC5", kijvalue = 2.33500000)
- type(kijdatadb), parameter **vdw265** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "H2", uid2 = "NC6", kijvalue = 2.45600000)
- type(kijdatadb), parameter **vdw266** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "H2", uid2 = "NC7", kijvalue = 2.63400000)
- type(kijdatadb), parameter **vdw267** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "H2", uid2 = "N2", kijvalue = 1.08000000)

- type(kijdatadb), parameter **vdw268** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "H2", uid2 = "CO", kijvalue = 1.08500000)
- type(kijdatadb), parameter **vdw269** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "H2", uid2 = "CO2", kijvalue = 1.62400000)
- type(kijdatadb), parameter **vdw270** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "O2", uid2 = "N2O", kijvalue = 1.05700000)
- type(kijdatadb), parameter **vdw271** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "CO", uid2 = "C1", kijvalue = 0.97400000)
- type(kijdatadb), parameter **vdw272** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "H2S", uid2 = "IC4", kijvalue = 0.94700000)
- type(kijdatadb), parameter **vdw273** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "N2O", uid2 = "C1", kijvalue = 1.01700000)
- type(kijdatadb), parameter **vdw274** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "H2O", uid2 = "CO2", kijvalue = 0.92000000)
- type(kijdatadb), parameter **vdw275** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "H2O", uid2 = "NH3", kijvalue = 1.15200000)
- type(kijdatadb), parameter **vdw276** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.1021/i260067a020", uid1 = "H2O", uid2 = "MEOH", kijvalue = 0.97900000)
- type(kijdatadb), parameter **vdw277** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "10.3303/CET1332311", uid1 = "CO2", uid2 = "AR", kijvalue = 0.99200000)
- type(kijdatadb), parameter **vdw278** = kijdatadb(eosid = "LK", mruleid = "vdW", ref = "Plocker1982(+)", bib_ref = "", uid1 = "CO2", uid2 = "O2", kijvalue = 1.03200000)
- type(kijdatadb), parameter **vdw279** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "H2S", kijvalue = 0.09300000)
- type(kijdatadb), parameter **vdw280** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "O2", kijvalue = 0.10200000)
- type(kijdatadb), parameter **vdw281** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NH3", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw282** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "N2O", kijvalue = 0.00700000)
- type(kijdatadb), parameter **vdw283** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "N2O4", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw284** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "C1", kijvalue = 0.09200000)
- type(kijdatadb), parameter **vdw285** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "C2", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw286** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "C2_1", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw287** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "C3", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw288** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "H2O", kijvalue = 0.07400000)
- type(kijdatadb), parameter **vdw289** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "IC4", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw290** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "IC5", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw291** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC10", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw292** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC11", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw293** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC4", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw294** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC5", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw295** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "H2S", kijvalue = 0.09900000)

- type(kijdatadb), parameter **vdw296** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "N2", kijvalue = -0.03600000)
- type(kijdatadb), parameter **vdw297** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "AR", kijvalue = 0.08600000)
- type(kijdatadb), parameter **vdw298** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "SO2", kijvalue = 0.07200000)
- type(kijdatadb), parameter **vdw299** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "H2", kijvalue = 0.10400000)
- type(kijdatadb), parameter **vdw300** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "CO", kijvalue = -0.06600000)
- type(kijdatadb), parameter **vdw301** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NH3", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw302** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2O", uid2 = "CO2", kijvalue = -0.06500000)
- type(kijdatadb), parameter **vdw303** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC4", kijvalue = 0.06000000)
- type(kijdatadb), parameter **vdw304** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC4", kijvalue = 0.06000000)
- type(kijdatadb), parameter **vdw305** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC5", kijvalue = 0.06000000)
- type(kijdatadb), parameter **vdw306** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC6", kijvalue = 0.05000000)
- type(kijdatadb), parameter **vdw307** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC7", kijvalue = 0.04000000)
- type(kijdatadb), parameter **vdw308** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC8", kijvalue = 0.04000000)
- type(kijdatadb), parameter **vdw309** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC9", kijvalue = 0.03000000)
- type(kijdatadb), parameter **vdw310** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MEOH", uid2 = "CO2", kijvalue = 0.01700000)
- type(kijdatadb), parameter **vdw311** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "C1", kijvalue = 0.03500000)
- type(kijdatadb), parameter **vdw312** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "C2", kijvalue = 0.04100000)
- type(kijdatadb), parameter **vdw313** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "C2_1", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw314** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "C3", kijvalue = 0.07600000)
- type(kijdatadb), parameter **vdw315** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "IC4", kijvalue = 0.09400000)
- type(kijdatadb), parameter **vdw316** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "IC5", kijvalue = 0.08700000)
- type(kijdatadb), parameter **vdw317** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC10", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw318** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC11", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw319** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC4", kijvalue = 0.07000000)
- type(kijdatadb), parameter **vdw320** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC5", kijvalue = 0.08800000)
- type(kijdatadb), parameter **vdw321** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC6", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw322** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC7", kijvalue = 0.14200000)
- type(kijdatadb), parameter **vdw323** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC8", kijvalue = 0.08000000)

- type(kijdatadb), parameter **vdw324** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC9", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw325** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "O2", kijvalue = -0.01400000)
- type(kijdatadb), parameter **vdw326** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R12", uid2 = "R11", kijvalue = 0.00540000)
- type(kijdatadb), parameter **vdw327** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R12", uid2 = "R114", kijvalue = 0.00150000)
- type(kijdatadb), parameter **vdw328** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R12", uid2 = "R152a", kijvalue = 0.08670000)
- type(kijdatadb), parameter **vdw329** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R13", uid2 = "R11", kijvalue = 0.02620000)
- type(kijdatadb), parameter **vdw330** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R13", uid2 = "R12", kijvalue = 0.02990000)
- type(kijdatadb), parameter **vdw331** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R14", uid2 = "R13", kijvalue = 0.03040000)
- type(kijdatadb), parameter **vdw332** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R14", uid2 = "R23", kijvalue = 0.10080000)
- type(kijdatadb), parameter **vdw333** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R11", kijvalue = 0.04660000)
- type(kijdatadb), parameter **vdw334** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R114", kijvalue = 0.03990000)
- type(kijdatadb), parameter **vdw335** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R12", kijvalue = 0.05640000)
- type(kijdatadb), parameter **vdw336** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R142b", kijvalue = 0.00570000)
- type(kijdatadb), parameter **vdw337** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R115", kijvalue = 0.08700000)
- type(kijdatadb), parameter **vdw338** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R23", uid2 = "R13", kijvalue = 0.10320000)
- type(kijdatadb), parameter **vdw339** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R218", uid2 = "R152a", kijvalue = 0.12000000)
- type(kijdatadb), parameter **vdw340** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R125", uid2 = "R143a", kijvalue = -0.01110000)
- type(kijdatadb), parameter **vdw341** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R125", uid2 = "R134a", kijvalue = -0.00240000)
- type(kijdatadb), parameter **vdw342** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R143a", uid2 = "R134a", kijvalue = 0.00130000)
- type(kijdatadb), parameter **vdw343** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R1234yf", uid2 = "R32", kijvalue = 0.03700000)
- type(kijdatadb), parameter **vdw344** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R1234yf", uid2 = "R125", kijvalue = 0.00400000)
- type(kijdatadb), parameter **vdw345** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R1234yf", uid2 = "R134a", kijvalue = 0.02000000)
- type(kijdatadb), parameter **vdw346** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R1234yf", uid2 = "CO2", kijvalue = 0.02000000)
- type(kijdatadb), parameter **vdw347** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "C3", kijvalue = 0.01400000)
- type(kijdatadb), parameter **vdw348** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "NC6", kijvalue = 0.04220000)
- type(kijdatadb), parameter **vdw349** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "C2", kijvalue = -0.00260000)
- type(kijdatadb), parameter **vdw350** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "H2O", kijvalue = 0.50000000)
- type(kijdatadb), parameter **vdw351** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "NC4", kijvalue = 0.01330000)

- type(kijdatadb), parameter **vdw352** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "IC4", kijvalue = 0.02560000)
- type(kijdatadb), parameter **vdw353** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "NC5", kijvalue = 0.02300000)
- type(kijdatadb), parameter **vdw354** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "NC7", kijvalue = 0.03520000)
- type(kijdatadb), parameter **vdw355** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "NC8", kijvalue = 0.04960000)
- type(kijdatadb), parameter **vdw356** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "IC5", kijvalue = -0.00560000)
- type(kijdatadb), parameter **vdw357** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "NC6", kijvalue = 0.00070000)
- type(kijdatadb), parameter **vdw358** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "C2", kijvalue = 0.00110000)
- type(kijdatadb), parameter **vdw359** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "H2S", kijvalue = 0.07500000)
- type(kijdatadb), parameter **vdw360** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "H2O", kijvalue = 0.48000000)
- type(kijdatadb), parameter **vdw361** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "NC4", kijvalue = 0.00330000)
- type(kijdatadb), parameter **vdw362** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "IC4", kijvalue = -0.00780000)
- type(kijdatadb), parameter **vdw363** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "NC5", kijvalue = 0.02670000)
- type(kijdatadb), parameter **vdw364** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "NC7", kijvalue = 0.00560000)
- type(kijdatadb), parameter **vdw365** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "NC8", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw366** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "IC5", kijvalue = 0.01110000)
- type(kijdatadb), parameter **vdw367** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "C2", kijvalue = -0.01000000)
- type(kijdatadb), parameter **vdw368** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "CO2", kijvalue = 0.11000000)
- type(kijdatadb), parameter **vdw369** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "H2O", kijvalue = 0.48000000)
- type(kijdatadb), parameter **vdw370** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "NC4", kijvalue = -0.00560000)
- type(kijdatadb), parameter **vdw371** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "IC4", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw372** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "NC5", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw373** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "NC7", kijvalue = -0.00780000)
- type(kijdatadb), parameter **vdw374** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "NC8", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw375** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "IC5", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw376** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "H2S", kijvalue = 0.08500000)
- type(kijdatadb), parameter **vdw377** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "H2O", kijvalue = 0.50000000)
- type(kijdatadb), parameter **vdw378** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "NC4", kijvalue = 0.00960000)
- type(kijdatadb), parameter **vdw379** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "IC4", kijvalue = -0.00670000)

- type(kijdatadb), parameter **vdw380** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "NC5", kijvalue = 0.00780000)
- type(kijdatadb), parameter **vdw381** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "NC7", kijvalue = 0.00740000)
- type(kijdatadb), parameter **vdw382** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "NC8", kijvalue = 0.01850000)
- type(kijdatadb), parameter **vdw383** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "IC5", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw384** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC7", kijvalue = 0.10000000)
- type(kijdatadb), parameter **vdw385** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC8", kijvalue = 0.10700000)
- type(kijdatadb), parameter **vdw386** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "H2O", kijvalue = 0.16400000)
- type(kijdatadb), parameter **vdw387** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "IC4", kijvalue = 0.06000000)
- type(kijdatadb), parameter **vdw388** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "IC5", kijvalue = 0.06000000)
- type(kijdatadb), parameter **vdw389** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2O", uid2 = "NC4", kijvalue = 0.48000000)
- type(kijdatadb), parameter **vdw390** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2O", uid2 = "IC4", kijvalue = 0.48000000)
- type(kijdatadb), parameter **vdw391** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2O", uid2 = "NC5", kijvalue = 0.48000000)
- type(kijdatadb), parameter **vdw392** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2O", uid2 = "NC7", kijvalue = 0.48000000)
- type(kijdatadb), parameter **vdw393** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2O", uid2 = "NC8", kijvalue = 0.48000000)
- type(kijdatadb), parameter **vdw394** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2O", uid2 = "IC5", kijvalue = 0.48000000)
- type(kijdatadb), parameter **vdw395** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "IC4", kijvalue = -0.00040000)
- type(kijdatadb), parameter **vdw396** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "NC5", kijvalue = 0.01740000)
- type(kijdatadb), parameter **vdw397** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "NC7", kijvalue = 0.00330000)
- type(kijdatadb), parameter **vdw398** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "NC8", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw399** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "IC5", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw400** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "NC5", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw401** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "NC7", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw402** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "NC8", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw403** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "IC5", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw404** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "NC7", kijvalue = 0.00740000)
- type(kijdatadb), parameter **vdw405** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "NC8", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw406** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "IC5", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw407** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC7", uid2 = "NC8", kijvalue = 0.00000000)

- type(kijdatadb), parameter **vdw408** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC7", uid2 = "IC5", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw409** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC8", uid2 = "IC5", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw410** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC5", uid2 = "NC5", kijvalue = 0.00000123)
- type(kijdatadb), parameter **vdw411** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC5", uid2 = "NC7", kijvalue = 0.00137330)
- type(kijdatadb), parameter **vdw412** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC5", uid2 = "NC8", kijvalue = 0.00276186)
- type(kijdatadb), parameter **vdw413** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "BENZENE", kijvalue = 0.08060000)
- type(kijdatadb), parameter **vdw414** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "TOLU", kijvalue = 0.09360000)
- type(kijdatadb), parameter **vdw415** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "EBZN", kijvalue = 0.10100000)
- type(kijdatadb), parameter **vdw416** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "MXYL", kijvalue = 0.08790000)
- type(kijdatadb), parameter **vdw417** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "OXYL", kijvalue = 0.09000000)
- type(kijdatadb), parameter **vdw418** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC9", kijvalue = 0.10100000)
- type(kijdatadb), parameter **vdw419** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "H2S", kijvalue = 0.16760000)
- type(kijdatadb), parameter **vdw420** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "BENZENE", kijvalue = 0.15970000)
- type(kijdatadb), parameter **vdw421** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "TOLU", kijvalue = 0.19320000)
- type(kijdatadb), parameter **vdw422** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "EBZN", kijvalue = 0.10000000)
- type(kijdatadb), parameter **vdw423** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "MXYL", kijvalue = 0.21690000)
- type(kijdatadb), parameter **vdw424** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "OXYL", kijvalue = 0.21400000)
- type(kijdatadb), parameter **vdw425** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "H2O", kijvalue = -0.31560000)
- type(kijdatadb), parameter **vdw426** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "C3_1", kijvalue = 0.06730000)
- type(kijdatadb), parameter **vdw427** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "BENZENE", kijvalue = 0.00900000)
- type(kijdatadb), parameter **vdw428** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "TOLU", kijvalue = 0.00810000)
- type(kijdatadb), parameter **vdw429** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "EBZN", kijvalue = 0.04500000)
- type(kijdatadb), parameter **vdw430** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "MXYL", kijvalue = 0.01710000)
- type(kijdatadb), parameter **vdw431** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "OXYL", kijvalue = -0.02310000)
- type(kijdatadb), parameter **vdw432** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC10", kijvalue = 0.04500000)
- type(kijdatadb), parameter **vdw433** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC11", kijvalue = 0.04500000)
- type(kijdatadb), parameter **vdw434** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "BENZENE", kijvalue = 0.04000000)
- type(kijdatadb), parameter **vdw435** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "TOLU", kijvalue = 0.06490000)

- type(kijdatadb), parameter **vdw436** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "EBZN", kijvalue = 0.02404000)
- type(kijdatadb), parameter **vdw437** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "MXYL", kijvalue = 0.04910000)
- type(kijdatadb), parameter **vdw438** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "OXYL", kijvalue = 0.05000000)
- type(kijdatadb), parameter **vdw439** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "NC9", kijvalue = 0.03893000)
- type(kijdatadb), parameter **vdw440** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "NC10", kijvalue = 0.04361000)
- type(kijdatadb), parameter **vdw441** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "NC11", kijvalue = 0.04799000)
- type(kijdatadb), parameter **vdw442** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "BENZENE", kijvalue = 0.02000000)
- type(kijdatadb), parameter **vdw443** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "TOLU", kijvalue = 0.03440000)
- type(kijdatadb), parameter **vdw444** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "EBZN", kijvalue = 0.01182000)
- type(kijdatadb), parameter **vdw445** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "MXYL", kijvalue = 0.02950000)
- type(kijdatadb), parameter **vdw446** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "OXYL", kijvalue = 0.03300000)
- type(kijdatadb), parameter **vdw447** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "NC9", kijvalue = 0.02302000)
- type(kijdatadb), parameter **vdw448** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "NC10", kijvalue = 0.02673000)
- type(kijdatadb), parameter **vdw449** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "NC11", kijvalue = 0.03026000)
- type(kijdatadb), parameter **vdw450** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "BENZENE", kijvalue = 0.02000000)
- type(kijdatadb), parameter **vdw451** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "TOLU", kijvalue = 0.03100000)
- type(kijdatadb), parameter **vdw452** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "EBZN", kijvalue = 0.00542000)
- type(kijdatadb), parameter **vdw453** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "MXYL", kijvalue = 0.03000000)
- type(kijdatadb), parameter **vdw454** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "OXYL", kijvalue = 0.03000000)
- type(kijdatadb), parameter **vdw455** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "NC9", kijvalue = 0.01370000)
- type(kijdatadb), parameter **vdw456** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "NC10", kijvalue = 0.01663000)
- type(kijdatadb), parameter **vdw457** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "NC11", kijvalue = 0.01948000)
- type(kijdatadb), parameter **vdw458** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "BENZENE", kijvalue = 0.00000180)
- type(kijdatadb), parameter **vdw459** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "TOLU", kijvalue = 0.00047000)
- type(kijdatadb), parameter **vdw460** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "EBZN", kijvalue = 0.00172000)
- type(kijdatadb), parameter **vdw461** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "MXYL", kijvalue = 0.00176000)
- type(kijdatadb), parameter **vdw462** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "OXYL", kijvalue = 0.00159000)
- type(kijdatadb), parameter **vdw463** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "NC9", kijvalue = 0.00725000)

- type(kijdatadb), parameter **vdw464** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "NC10", kijvalue = 0.00945000)
- type(kijdatadb), parameter **vdw465** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "NC11", kijvalue = 0.01164000)
- type(kijdatadb), parameter **vdw466** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "BENZENE", kijvalue = 0.00001000)
- type(kijdatadb), parameter **vdw467** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "TOLU", kijvalue = 0.00064000)
- type(kijdatadb), parameter **vdw468** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "EBZN", kijvalue = 0.00203000)
- type(kijdatadb), parameter **vdw469** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "MXYL", kijvalue = 0.00208000)
- type(kijdatadb), parameter **vdw470** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "OXYL", kijvalue = 0.00190000)
- type(kijdatadb), parameter **vdw471** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "NC9", kijvalue = 0.00788000)
- type(kijdatadb), parameter **vdw472** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "NC10", kijvalue = 0.01016000)
- type(kijdatadb), parameter **vdw473** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "NC11", kijvalue = 0.01243000)
- type(kijdatadb), parameter **vdw474** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC5", uid2 = "NC5", kijvalue = 0.00000130)
- type(kijdatadb), parameter **vdw475** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC5", uid2 = "BENZENE", kijvalue = 0.00040000)
- type(kijdatadb), parameter **vdw476** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC5", uid2 = "TOLU", kijvalue = 0.00001000)
- type(kijdatadb), parameter **vdw477** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC5", uid2 = "EBZN", kijvalue = 0.00052000)
- type(kijdatadb), parameter **vdw478** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC5", uid2 = "MXYL", kijvalue = 0.00055000)
- type(kijdatadb), parameter **vdw479** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC5", uid2 = "OXYL", kijvalue = 0.00046000)
- type(kijdatadb), parameter **vdw480** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC5", uid2 = "NC9", kijvalue = 0.00445000)
- type(kijdatadb), parameter **vdw481** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC5", uid2 = "NC10", kijvalue = 0.00621000)
- type(kijdatadb), parameter **vdw482** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC5", uid2 = "NC11", kijvalue = 0.00801000)
- type(kijdatadb), parameter **vdw483** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC5", uid2 = "BENZENE", kijvalue = 0.01600000)
- type(kijdatadb), parameter **vdw484** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC5", uid2 = "TOLU", kijvalue = 0.00000360)
- type(kijdatadb), parameter **vdw485** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC5", uid2 = "EBZN", kijvalue = 0.00047000)
- type(kijdatadb), parameter **vdw486** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC5", uid2 = "MXYL", kijvalue = 0.00050000)
- type(kijdatadb), parameter **vdw487** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC5", uid2 = "OXYL", kijvalue = 0.00041000)
- type(kijdatadb), parameter **vdw488** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC5", uid2 = "NC7", kijvalue = 0.00137000)
- type(kijdatadb), parameter **vdw489** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC5", uid2 = "NC8", kijvalue = 0.00276000)
- type(kijdatadb), parameter **vdw490** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC5", uid2 = "NC9", kijvalue = 0.00430000)
- type(kijdatadb), parameter **vdw491** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC5", uid2 = "NC10", kijvalue = 0.00603000)

- type(kijdatadb), parameter **vdw492** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC5", uid2 = "NC11", kijvalue = 0.00781000)
- type(kijdatadb), parameter **vdw493** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "BENZENE", uid2 = "TOLU", kijvalue = 0.00053000)
- type(kijdatadb), parameter **vdw494** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "BENZENE", uid2 = "EBZN", kijvalue = 0.00183000)
- type(kijdatadb), parameter **vdw495** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "BENZENE", uid2 = "MXYL", kijvalue = 0.00188000)
- type(kijdatadb), parameter **vdw496** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "BENZENE", uid2 = "OXYL", kijvalue = 0.00170000)
- type(kijdatadb), parameter **vdw497** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "BENZENE", uid2 = "NC6", kijvalue = 0.00700000)
- type(kijdatadb), parameter **vdw498** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "BENZENE", uid2 = "NC7", kijvalue = -0.00200000)
- type(kijdatadb), parameter **vdw499** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "BENZENE", uid2 = "NC8", kijvalue = 0.00300000)
- type(kijdatadb), parameter **vdw500** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "BENZENE", uid2 = "NC9", kijvalue = 0.00749000)
- type(kijdatadb), parameter **vdw501** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "BENZENE", uid2 = "NC10", kijvalue = 0.01000000)
- type(kijdatadb), parameter **vdw502** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "BENZENE", uid2 = "NC11", kijvalue = 0.01193000)
- type(kijdatadb), parameter **vdw503** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "BENZENE", uid2 = "H2O", kijvalue = 0.50000000)
- type(kijdatadb), parameter **vdw504** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "TOLU", uid2 = "EBZN", kijvalue = 0.00039000)
- type(kijdatadb), parameter **vdw505** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "TOLU", uid2 = "MXYL", kijvalue = 0.00042000)
- type(kijdatadb), parameter **vdw506** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "TOLU", uid2 = "OXYL", kijvalue = 0.00034000)
- type(kijdatadb), parameter **vdw507** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "TOLU", uid2 = "NC6", kijvalue = 0.00032000)
- type(kijdatadb), parameter **vdw508** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "TOLU", uid2 = "NC7", kijvalue = 0.00600000)
- type(kijdatadb), parameter **vdw509** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "TOLU", uid2 = "NC8", kijvalue = 0.01000000)
- type(kijdatadb), parameter **vdw510** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "TOLU", uid2 = "NC9", kijvalue = 0.00406000)
- type(kijdatadb), parameter **vdw511** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "TOLU", uid2 = "NC10", kijvalue = 0.01000000)
- type(kijdatadb), parameter **vdw512** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "TOLU", uid2 = "NC11", kijvalue = 0.00749000)
- type(kijdatadb), parameter **vdw513** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "TOLU", uid2 = "H2O", kijvalue = 0.50000000)
- type(kijdatadb), parameter **vdw514** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "EBZN", uid2 = "MXYL", kijvalue = 0.00000020)
- type(kijdatadb), parameter **vdw515** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "EBZN", uid2 = "OXYL", kijvalue = 0.00000020)
- type(kijdatadb), parameter **vdw516** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "EBZN", uid2 = "NC6", kijvalue = 0.00000400)
- type(kijdatadb), parameter **vdw517** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "EBZN", uid2 = "NC7", kijvalue = 0.00024000)
- type(kijdatadb), parameter **vdw518** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "EBZN", uid2 = "NC8", kijvalue = -0.00180000)
- type(kijdatadb), parameter **vdw519** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "EBZN", uid2 = "NC9", kijvalue = 0.00193000)

- type(kijdatadb), parameter **vdw520** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "EBZN", uid2 = "NC10", kijvalue = 0.00314000)
- type(kijdatadb), parameter **vdw521** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "EBZN", uid2 = "NC11", kijvalue = 0.00446000)
- type(kijdatadb), parameter **vdw522** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "EBZN", uid2 = "H2O", kijvalue = 0.50000000)
- type(kijdatadb), parameter **vdw523** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MXYL", uid2 = "OXYL", kijvalue = 0.00000400)
- type(kijdatadb), parameter **vdw524** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MXYL", uid2 = "NC6", kijvalue = 0.00001000)
- type(kijdatadb), parameter **vdw525** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MXYL", uid2 = "NC7", kijvalue = 0.00022000)
- type(kijdatadb), parameter **vdw526** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MXYL", uid2 = "NC8", kijvalue = 0.00092000)
- type(kijdatadb), parameter **vdw527** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MXYL", uid2 = "NC9", kijvalue = 0.00188000)
- type(kijdatadb), parameter **vdw528** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MXYL", uid2 = "NC10", kijvalue = 0.00308000)
- type(kijdatadb), parameter **vdw529** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MXYL", uid2 = "NC11", kijvalue = 0.00439000)
- type(kijdatadb), parameter **vdw530** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MXYL", uid2 = "H2O", kijvalue = 0.50000000)
- type(kijdatadb), parameter **vdw531** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "OXYL", uid2 = "NC6", kijvalue = 0.00000020)
- type(kijdatadb), parameter **vdw532** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "OXYL", uid2 = "NC7", kijvalue = 0.00029000)
- type(kijdatadb), parameter **vdw533** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "OXYL", uid2 = "NC8", kijvalue = 0.00105000)
- type(kijdatadb), parameter **vdw534** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "OXYL", uid2 = "NC9", kijvalue = 0.00207000)
- type(kijdatadb), parameter **vdw535** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "OXYL", uid2 = "NC10", kijvalue = 0.00331000)
- type(kijdatadb), parameter **vdw536** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "OXYL", uid2 = "NC11", kijvalue = 0.00467000)
- type(kijdatadb), parameter **vdw537** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "OXYL", uid2 = "H2O", kijvalue = 0.50000000)
- type(kijdatadb), parameter **vdw538** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "NC9", kijvalue = 0.00210000)
- type(kijdatadb), parameter **vdw539** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "NC10", kijvalue = 0.00335000)
- type(kijdatadb), parameter **vdw540** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "NC11", kijvalue = 0.00472000)
- type(kijdatadb), parameter **vdw541** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC7", uid2 = "NC9", kijvalue = 0.00082000)
- type(kijdatadb), parameter **vdw542** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC7", uid2 = "NC10", kijvalue = 0.00166000)
- type(kijdatadb), parameter **vdw543** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC7", uid2 = "NC11", kijvalue = 0.00266000)
- type(kijdatadb), parameter **vdw544** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC8", uid2 = "NC9", kijvalue = 0.00017000)
- type(kijdatadb), parameter **vdw545** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC8", uid2 = "NC10", kijvalue = 0.00064000)
- type(kijdatadb), parameter **vdw546** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC8", uid2 = "NC11", kijvalue = 0.00130000)
- type(kijdatadb), parameter **vdw547** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC9", uid2 = "NC10", kijvalue = 0.00130000)

- type(kijdatadb), parameter **vdw548** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC9", uid2 = "NC11", kijvalue = 0.00053000)
- type(kijdatadb), parameter **vdw549** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC9", uid2 = "H2O", kijvalue = 0.50000000)
- type(kijdatadb), parameter **vdw550** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC10", uid2 = "NC11", kijvalue = 0.00012000)
- type(kijdatadb), parameter **vdw551** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC10", uid2 = "H2O", kijvalue = 0.50000000)
- type(kijdatadb), parameter **vdw552** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC11", uid2 = "H2O", kijvalue = 0.50000000)
- type(kijdatadb), parameter **vdw553** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "QuantumCubic", bib_ref = "10.1016/j.fluid.2020.112790", uid1 = "H2", uid2 = "D2", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw554** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "QuantumCubic", bib_ref = "10.1016/j.fluid.2020.112790", uid1 = "H2", uid2 = "NE", kijvalue = 0.18000000)
- type(kijdatadb), parameter **vdw555** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "QuantumCubic", bib_ref = "10.1016/j.fluid.2020.112790", uid1 = "H2", uid2 = "HE", kijvalue = 0.17000000)
- type(kijdatadb), parameter **vdw556** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "QuantumCubic", bib_ref = "10.1016/j.fluid.2020.112790", uid1 = "HE", uid2 = "NE", kijvalue = -0.17000000)
- type(kijdatadb), parameter **vdw557** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "QuantumCubic", bib_ref = "10.1016/j.fluid.2020.112790", uid1 = "HE", uid2 = "D2", kijvalue = 0.45000000)
- type(kijdatadb), parameter **vdw558** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "QuantumCubic", bib_ref = "10.1016/j.fluid.2020.112790", uid1 = "NE", uid2 = "D2", kijvalue = 0.18000000)
- type(kijdatadb), parameter **vdw559** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2O2", uid2 = "H2O", kijvalue = -0.05000000)
- type(kijdatadb), parameter **vdw560** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "tcPR-ENGINEERING", bib_ref = "", uid1 = "H2O2", uid2 = "O2", kijvalue = -0.75000000)
- type(kijdatadb), parameter **vdw561** = kijdatadb(eosid = "PR", mruleid = "vdW", ref = "tcPR-ENGINEERING", bib_ref = "", uid1 = "H2O", uid2 = "O2", kijvalue = -0.26000000)
- type(kijdatadb), parameter **vdw562** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "C1", uid2 = "C2", kijvalue = 0.00500000)
- type(kijdatadb), parameter **vdw563** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "C1", uid2 = "IC4", kijvalue = -0.00800000)
- type(kijdatadb), parameter **vdw564** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "C1", uid2 = "NC4", kijvalue = 0.01500000)
- type(kijdatadb), parameter **vdw565** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "C1", uid2 = "NC5", kijvalue = 0.02000000)
- type(kijdatadb), parameter **vdw566** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "C1", uid2 = "NC6", kijvalue = 0.01000000)
- type(kijdatadb), parameter **vdw567** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "C2", uid2 = "C3", kijvalue = 0.00600000)
- type(kijdatadb), parameter **vdw568** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "C2", uid2 = "PRLN", kijvalue = 0.00400000)
- type(kijdatadb), parameter **vdw569** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "C2", uid2 = "NC4", kijvalue = 0.00100000)
- type(kijdatadb), parameter **vdw570** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "C2", uid2 = "NC5", kijvalue = -0.00100000)
- type(kijdatadb), parameter **vdw571** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "C2", uid2 = "NC7", kijvalue = -0.01500000)
- type(kijdatadb), parameter **vdw572** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "C2_1", uid2 = "C1", kijvalue = 0.02600000)
- type(kijdatadb), parameter **vdw573** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "C2_1", uid2 = "C2", kijvalue = 0.01800000)
- type(kijdatadb), parameter **vdw574** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "C2_1", uid2 = "C3", kijvalue = 0.01900000)
- type(kijdatadb), parameter **vdw575** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "C2_1", uid2 = "NC4", kijvalue = 0.06200000)

- type(kijdatadb), parameter **vdw576** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "C3", uid2 = "NC4", kijvalue = 0.01300000)
- type(kijdatadb), parameter **vdw577** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "C3", uid2 = "NC5", kijvalue = 0.01300000)
- type(kijdatadb), parameter **vdw578** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "C3", uid2 = "IC5", kijvalue = 0.02100000)
- type(kijdatadb), parameter **vdw579** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "PRLN", uid2 = "C3", kijvalue = 0.03000000)
- type(kijdatadb), parameter **vdw580** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "NC4", uid2 = "NC10", kijvalue = 0.00500000)
- type(kijdatadb), parameter **vdw581** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "IC4", uid2 = "NC4", kijvalue = -0.00300000)
- type(kijdatadb), parameter **vdw582** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "CO2", uid2 = "C1", kijvalue = 0.09300000)
- type(kijdatadb), parameter **vdw583** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "CO2", uid2 = "C2", kijvalue = 0.12800000)
- type(kijdatadb), parameter **vdw584** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "CO2", uid2 = "C2_1", kijvalue = 0.05700000)
- type(kijdatadb), parameter **vdw585** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "CO2", uid2 = "C3", kijvalue = 0.13100000)
- type(kijdatadb), parameter **vdw586** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "CO2", uid2 = "NC4", kijvalue = 0.10900000)
- type(kijdatadb), parameter **vdw587** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "CO2", uid2 = "IC4", kijvalue = 0.12700000)
- type(kijdatadb), parameter **vdw588** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "CO2", uid2 = "NC5", kijvalue = 0.13500000)
- type(kijdatadb), parameter **vdw589** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "H2S", uid2 = "C1", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw590** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "H2S", uid2 = "C2", kijvalue = 0.08900000)
- type(kijdatadb), parameter **vdw591** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "H2S", uid2 = "IC4", kijvalue = 0.04600000)
- type(kijdatadb), parameter **vdw592** = kijdatadb(eosid = "PT", mruleid = "vdW", ref = "Patel1982", bib_ref = "10.1016/0009-2509(82)80099-7", uid1 = "H2S", uid2 = "NC7", kijvalue = 0.05300000)
- type(kijdatadb), parameter **vdw593** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "C2", kijvalue = -0.00780000)
- type(kijdatadb), parameter **vdw594** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "C2_1", kijvalue = 0.02000000)
- type(kijdatadb), parameter **vdw595** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "C3", kijvalue = 0.00900000)
- type(kijdatadb), parameter **vdw596** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "IC4", kijvalue = 0.02410000)
- type(kijdatadb), parameter **vdw597** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "NC4", kijvalue = 0.00560000)
- type(kijdatadb), parameter **vdw598** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "NC5", kijvalue = 0.01900000)
- type(kijdatadb), parameter **vdw599** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "H2S", kijvalue = 0.09100000)
- type(kijdatadb), parameter **vdw600** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "C2_1", kijvalue = 0.01120000)
- type(kijdatadb), parameter **vdw601** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "C3", kijvalue = -0.00220000)
- type(kijdatadb), parameter **vdw602** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "IC4", kijvalue = -0.01000000)
- type(kijdatadb), parameter **vdw603** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "NC4", kijvalue = 0.00670000)

- type(kijdatadb), parameter **vdw604** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "NC5", kijvalue = 0.00560000)
- type(kijdatadb), parameter **vdw605** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "C2_1", kijvalue = 0.10000000)
- type(kijdatadb), parameter **vdw606** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "IC4", kijvalue = -0.01000000)
- type(kijdatadb), parameter **vdw607** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "NC5", kijvalue = 0.02300000)
- type(kijdatadb), parameter **vdw608** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "NC4", kijvalue = 0.00110000)
- type(kijdatadb), parameter **vdw609** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "NC5", kijvalue = 0.02040000)
- type(kijdatadb), parameter **vdw610** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2O", uid2 = "C1", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw611** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2O", uid2 = "MEG", kijvalue = -0.06300000)
- type(kijdatadb), parameter **vdw612** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "MEG", kijvalue = 0.05000000)
- type(kijdatadb), parameter **vdw613** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "O2", kijvalue = 0.10400000)
- type(kijdatadb), parameter **vdw614** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NH3", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw615** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "N2O", kijvalue = 0.00400000)
- type(kijdatadb), parameter **vdw616** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "N2O4", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw617** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "CO", kijvalue = -0.08000000)
- type(kijdatadb), parameter **vdw618** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "C1", kijvalue = 0.09500000)
- type(kijdatadb), parameter **vdw619** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "C2", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw620** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "C2_1", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw621** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "C3", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw622** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "H2O", kijvalue = 0.07400000)
- type(kijdatadb), parameter **vdw623** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "IC4", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw624** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "IC5", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw625** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC10", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw626** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC11", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw627** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC4", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw628** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC5", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw629** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "H2S", kijvalue = 0.10000000)
- type(kijdatadb), parameter **vdw630** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "N2", kijvalue = -0.05100000)
- type(kijdatadb), parameter **vdw631** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "AR", kijvalue = 0.08800000)

- type(kijdatadb), parameter **vdw632** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "SO2", kijvalue = 0.07100000)
- type(kijdatadb), parameter **vdw633** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "H2", kijvalue = 0.00900000)
- type(kijdatadb), parameter **vdw634** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC4", kijvalue = 0.06000000)
- type(kijdatadb), parameter **vdw635** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC5", kijvalue = 0.06000000)
- type(kijdatadb), parameter **vdw636** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC6", kijvalue = 0.05000000)
- type(kijdatadb), parameter **vdw637** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC7", kijvalue = 0.04000000)
- type(kijdatadb), parameter **vdw638** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC8", kijvalue = 0.04000000)
- type(kijdatadb), parameter **vdw639** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "NC9", kijvalue = 0.03000000)
- type(kijdatadb), parameter **vdw640** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "MEOH", uid2 = "CO2", kijvalue = 0.01700000)
- type(kijdatadb), parameter **vdw641** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "C1", kijvalue = 0.03400000)
- type(kijdatadb), parameter **vdw642** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "C2", kijvalue = 0.06000000)
- type(kijdatadb), parameter **vdw643** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "C2_1", kijvalue = 0.07500000)
- type(kijdatadb), parameter **vdw644** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "C3", kijvalue = 0.09000000)
- type(kijdatadb), parameter **vdw645** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "IC4", kijvalue = 0.11300000)
- type(kijdatadb), parameter **vdw646** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "IC5", kijvalue = 0.08700000)
- type(kijdatadb), parameter **vdw647** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC10", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw648** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC11", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw649** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC4", kijvalue = 0.11300000)
- type(kijdatadb), parameter **vdw650** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC5", kijvalue = 0.14000000)
- type(kijdatadb), parameter **vdw651** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC6", kijvalue = 0.15000000)
- type(kijdatadb), parameter **vdw652** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC7", kijvalue = 0.14200000)
- type(kijdatadb), parameter **vdw653** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC8", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw654** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "NC9", kijvalue = 0.08000000)
- type(kijdatadb), parameter **vdw655** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "O2", kijvalue = -0.01100000)
- type(kijdatadb), parameter **vdw656** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R12", uid2 = "R11", kijvalue = 0.00540000)
- type(kijdatadb), parameter **vdw657** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R12", uid2 = "R114", kijvalue = 0.00150000)
- type(kijdatadb), parameter **vdw658** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R12", uid2 = "R152a", kijvalue = 0.08670000)
- type(kijdatadb), parameter **vdw659** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R13", uid2 = "R11", kijvalue = 0.02620000)

- type(kijdatadb), parameter **vdw660** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R13", uid2 = "R12", kijvalue = 0.02990000)
- type(kijdatadb), parameter **vdw661** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R14", uid2 = "R13", kijvalue = 0.03040000)
- type(kijdatadb), parameter **vdw662** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R14", uid2 = "R23", kijvalue = 0.10080000)
- type(kijdatadb), parameter **vdw663** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R11", kijvalue = 0.04660000)
- type(kijdatadb), parameter **vdw664** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R114", kijvalue = 0.03990000)
- type(kijdatadb), parameter **vdw665** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R12", kijvalue = 0.05640000)
- type(kijdatadb), parameter **vdw666** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R142b", kijvalue = 0.00570000)
- type(kijdatadb), parameter **vdw667** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R22", uid2 = "R115", kijvalue = 0.08900000)
- type(kijdatadb), parameter **vdw668** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R23", uid2 = "R13", kijvalue = 0.10320000)
- type(kijdatadb), parameter **vdw669** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R218", uid2 = "R152a", kijvalue = 0.12000000)
- type(kijdatadb), parameter **vdw670** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R125", uid2 = "R143a", kijvalue = -0.01110000)
- type(kijdatadb), parameter **vdw671** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R125", uid2 = "R134a", kijvalue = -0.00240000)
- type(kijdatadb), parameter **vdw672** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "R143a", uid2 = "R134a", kijvalue = 0.00130000)
- type(kijdatadb), parameter **vdw673** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "NC6", kijvalue = 0.03740000)
- type(kijdatadb), parameter **vdw674** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "NC7", kijvalue = 0.03070000)
- type(kijdatadb), parameter **vdw675** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "NC8", kijvalue = 0.04480000)
- type(kijdatadb), parameter **vdw676** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C1", uid2 = "IC5", kijvalue = -0.00780000)
- type(kijdatadb), parameter **vdw677** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "NC6", kijvalue = -0.00220000)
- type(kijdatadb), parameter **vdw678** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "H2S", kijvalue = 0.07600000)
- type(kijdatadb), parameter **vdw679** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "H2O", kijvalue = 0.53000000)
- type(kijdatadb), parameter **vdw680** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "NC4", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw681** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "NC7", kijvalue = 0.00440000)
- type(kijdatadb), parameter **vdw682** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "NC8", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw683** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C3", uid2 = "IC5", kijvalue = 0.00780000)
- type(kijdatadb), parameter **vdw684** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "C2", kijvalue = -0.01560000)
- type(kijdatadb), parameter **vdw685** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "CO2", kijvalue = 0.11200000)
- type(kijdatadb), parameter **vdw686** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "H2O", kijvalue = 0.50000000)
- type(kijdatadb), parameter **vdw687** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "NC4", kijvalue = -0.01110000)

- type(kijdatadb), parameter **vdw688** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "IC4", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw689** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "NC5", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw690** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "NC7", kijvalue = -0.00110000)
- type(kijdatadb), parameter **vdw691** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "NC8", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw692** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC6", uid2 = "IC5", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw693** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "H2S", kijvalue = 0.08500000)
- type(kijdatadb), parameter **vdw694** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "H2O", kijvalue = 0.55000000)
- type(kijdatadb), parameter **vdw695** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "NC7", kijvalue = 0.00410000)
- type(kijdatadb), parameter **vdw696** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "NC8", kijvalue = 0.01700000)
- type(kijdatadb), parameter **vdw697** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "C2", uid2 = "IC5", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw698** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC7", kijvalue = 0.11000000)
- type(kijdatadb), parameter **vdw699** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "CO2", uid2 = "NC8", kijvalue = 0.12000000)
- type(kijdatadb), parameter **vdw700** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "N2", kijvalue = 0.17000000)
- type(kijdatadb), parameter **vdw701** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "H2O", kijvalue = 0.13500000)
- type(kijdatadb), parameter **vdw702** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "IC4", kijvalue = 0.06000000)
- type(kijdatadb), parameter **vdw703** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2S", uid2 = "IC5", kijvalue = 0.06500000)
- type(kijdatadb), parameter **vdw704** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "N2", uid2 = "H2O", kijvalue = 0.53000000)
- type(kijdatadb), parameter **vdw705** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2O", uid2 = "NC4", kijvalue = 0.52000000)
- type(kijdatadb), parameter **vdw706** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2O", uid2 = "IC4", kijvalue = 0.52000000)
- type(kijdatadb), parameter **vdw707** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2O", uid2 = "NC5", kijvalue = 0.50000000)
- type(kijdatadb), parameter **vdw708** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2O", uid2 = "NC7", kijvalue = 0.50000000)
- type(kijdatadb), parameter **vdw709** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2O", uid2 = "NC8", kijvalue = 0.50000000)
- type(kijdatadb), parameter **vdw710** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "H2O", uid2 = "IC5", kijvalue = 0.50000000)
- type(kijdatadb), parameter **vdw711** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "NC7", kijvalue = -0.00040000)
- type(kijdatadb), parameter **vdw712** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "NC8", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw713** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC4", uid2 = "IC5", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw714** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "NC5", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw715** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "NC7", kijvalue = 0.00000000)

- type(kijdatadb), parameter **vdw716** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "NC8", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw717** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "IC4", uid2 = "IC5", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw718** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC5", uid2 = "NC7", kijvalue = 0.00190000)
- type(kijdatadb), parameter **vdw719** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC5", uid2 = "NC8", kijvalue = -0.00220000)
- type(kijdatadb), parameter **vdw720** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC5", uid2 = "IC5", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw721** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC7", uid2 = "NC8", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw722** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC7", uid2 = "IC5", kijvalue = 0.00000000)
- type(kijdatadb), parameter **vdw723** = kijdatadb(eosid = "SRK", mruleid = "vdW", ref = "Default", bib_ref = "", uid1 = "NC8", uid2 = "IC5", kijvalue = 0.00000000)
- type(intergedatadb), parameter **ge1** = interGEdatadb(eosid = "PR", mruleid = "HV1", ref = "Hemmingsen2011", bib_ref = "10.1016/j.fluid.2011.05.010", uid1 = "H2O", uid2 = "MEG", kijvalue = -0.↩ 06500000, correlation = 1, alphaijvalue = (/4.00000000e-01, 4.00000000e-01/), polyij = (/0.00000000e+00, 2.18300000e+02, 0.0/), polyji = (/0.00000000e+00, 7.20200000e+01, 0.0/))
- type(intergedatadb), parameter **ge2** = interGEdatadb(eosid = "PR", mruleid = "NRTL", ref = "Dicko2012", bib_ref = "10.1021/je300111m", uid1 = "H2S", uid2 = "C3", kijvalue = 0.07500000, correlation = 1, alphaijvalue = (/3.00000000e-01, 3.00000000e-01/), polyij = (/3.45712000e+02, 0.00000000e+00, 0.00000000e+00/), polyji = (/-2.93377000e+01, 0.00000000e+00, 0.00000000e+00/))
- type(intergedatadb), parameter **ge3** = interGEdatadb(eosid = "SRK", mruleid = "HV1", ref = "Default", bib_↩ ref = "10.1205/cherd05023", uid1 = "C1", uid2 = "H2O", kijvalue = 0.52000000, correlation = 1, alphaijvalue = (/1.50000000e-01, 1.50000000e-01/), polyij = (/5.03570000e+03, -7.15000000e+00, 0.0/), polyji = (/-1.↩ 62800000e+02, 2.16000000e+00, 0.0/))
- type(intergedatadb), parameter **ge4** = interGEdatadb(eosid = "SRK", mruleid = "HV2", ref = "Default", bib_ref = "10.1205/cherd05023", uid1 = "C1", uid2 = "H2O", kijvalue = 0.52000000, correlation = 1, alphaijvalue = (/1.50000000e-01, 1.50000000e-01/), polyij = (/8.40400000e+03, -2.80310000e+01, 3.14200000e-02/), polyji = (/-1.14900000e+03, 8.01900000e+00, -8.61000000e-03/))
- type(intergedatadb), parameter **ge5** = interGEdatadb(eosid = "SRK", mruleid = "HV1", ref = "Default", bib_ref = "10.1016/j.fluid.2006.08.021", uid1 = "C1", uid2 = "MEG", kijvalue = 0.13400000, correlation = 1, alphaijvalue = (/7.00000000e-02, 7.00000000e-02/), polyij = (/2.27400000e+03, 0.00000000e+00, 0.0/), polyji = (/1.81000000e+02, 0.00000000e+00, 0.0/))
- type(intergedatadb), parameter **ge6** = interGEdatadb(eosid = "SRK", mruleid = "HV2", ref = "Default", bib_ref = "10.1016/j.fluid.2006.08.021", uid1 = "C1", uid2 = "MEG", kijvalue = 0.13400000, correlation = 1, alphaijvalue = (/7.00000000e-02, 7.00000000e-02/), polyij = (/2.27400000e+03, 0.00000000e+00, 0.↩ 00000000e+00/), polyji = (/1.81000000e+02, 0.00000000e+00, 0.00000000e+00/))
- type(intergedatadb), parameter **ge7** = interGEdatadb(eosid = "SRK", mruleid = "HV1", ref = "Default", bib_ref = "10.1205/cherd05023", uid1 = "CO2", uid2 = "H2O", kijvalue = 0.19300000, correlation = 1, alphaijvalue = (/3.00000000e-02, 3.00000000e-02/), polyij = (/6.56300000e+03, -5.12000000e+00, 0.0/), polyji = (/-3.↩ 74100000e+03, 1.55000000e+00, 0.0/))
- type(intergedatadb), parameter **ge8** = interGEdatadb(eosid = "SRK", mruleid = "HV2", ref = "Default", bib_ref = "10.1205/cherd05023", uid1 = "CO2", uid2 = "H2O", kijvalue = 0.19300000, correlation = 1, alphaijvalue = (/3.00000000e-02, 3.00000000e-02/), polyij = (/5.85839000e+03, 2.41120000e+00, -1.71300000e-02/), polyji = (/-1.23741000e+03, -1.55058000e+01, 2.89600000e-02/))
- type(intergedatadb), parameter **ge9** = interGEdatadb(eosid = "SRK", mruleid = "HV1", ref = "Default", bib_ref = "10.1016/j.fluid.2006.08.021", uid1 = "H2O", uid2 = "MEG", kijvalue = -0.06300000, correlation = 1, alphaijvalue = (/9.50000000e-01, 9.50000000e-01/), polyij = (/5.90000000e+01, 0.00000000e+00, 0.0/), polyji = (/1.05000000e+02, 0.00000000e+00, 0.0/))
- type(intergedatadb), parameter **ge10** = interGEdatadb(eosid = "SRK", mruleid = "HV2", ref = "Default", bib↩ _ref = "10.1016/j.fluid.2006.08.021", uid1 = "H2O", uid2 = "MEG", kijvalue = -0.06300000, correlation = 1, alphaijvalue = (/9.50000000e-01, 9.50000000e-01/), polyij = (/5.90000000e+01, 0.00000000e+00, 0.↩ 00000000e+00/), polyji = (/1.05000000e+02, 0.00000000e+00, 0.00000000e+00/))

- type(intergedatadb), parameter **ge11** = interGEdatadb(eosid = "SRK", mruleid = "HV2", ref = "Default/Maribo-Mogensen", bib_ref = "10.1002/aic.14829", uid1 = "H2O", uid2 = "NA+", kijvalue = 0.00000000, correlation = 2, alphaijvalue = (/0.00000000e+00, 0.00000000e+00/), polyij = (/-2.23150000e+02, 1.57300000e+03, 3.↩ 40000000e+02/), polyji = (/-2.23150000e+02, 1.57300000e+03, 3.40000000e+02/))
- type(intergedatadb), parameter **ge12** = interGEdatadb(eosid = "SRK", mruleid = "HV2", ref = "Default/Maribo-Mogensen", bib_ref = "10.1002/aic.14829", uid1 = "H2O", uid2 = "CL-", kijvalue = 0.00000000, correlation = 2, alphaijvalue = (/0.00000000e+00, 0.00000000e+00/), polyij = (/-2.23150000e+02, 1.57300000e+03, 3.↩ 40000000e+02/), polyji = (/-2.23150000e+02, 1.57300000e+03, 3.40000000e+02/))
- type(intergedatadb), parameter **ge13** = interGEdatadb(eosid = "SRK", mruleid = "HV2", ref = "Default/Maribo-Mogensen", bib_ref = "10.1002/aic.14829", uid1 = "MEOH", uid2 = "NA+", kijvalue = 0.00000000, correlation = 2, alphaijvalue = (/0.00000000e+00, 0.00000000e+00/), polyij = (/3.22400000e+02, 0.00000000e+00, 1.↩ 00000000e+00/), polyji = (/3.22400000e+02, 0.00000000e+00, 1.00000000e+00/))
- type(intergedatadb), parameter **ge14** = interGEdatadb(eosid = "SRK", mruleid = "HV2", ref = "Default/Maribo-Mogensen", bib_ref = "10.1002/aic.14829", uid1 = "MEOH", uid2 = "CL-", kijvalue = 0.00000000, correlation = 2, alphaijvalue = (/0.00000000e+00, 0.00000000e+00/), polyij = (/3.22400000e+02, 0.00000000e+00, 1.↩ 00000000e+00/), polyji = (/3.22400000e+02, 0.00000000e+00, 1.00000000e+00/))
- type(intergedatadb), parameter **ge15** = interGEdatadb(eosid = "SRK", mruleid = "HV2", ref = "Default/Maribo-Mogensen", bib_ref = "10.1002/aic.14829", uid1 = "NA+", uid2 = "CL-", kijvalue = 0.00000000, correlation = 2, alphaijvalue = (/0.00000000e+00, 0.00000000e+00/), polyij = (/0.00000000e+00, 0.00000000e+00, 1.↩ 00000000e+00/), polyji = (/0.00000000e+00, 0.00000000e+00, 1.00000000e+00/))
- type(intergedatadb), parameter **ge16** = interGEdatadb(eosid = "SRK", mruleid = "HV2", ref = "Default/Maribo-Mogensen", bib_ref = "10.1002/aic.14829", uid1 = "CO2", uid2 = "NA+", kijvalue = 0.00000000, correlation = 2, alphaijvalue = (/0.00000000e+00, 0.00000000e+00/), polyij = (/7.24800000e+02, 0.00000000e+00, 1.↩ 00000000e+00/), polyji = (/7.24800000e+02, 0.00000000e+00, 1.00000000e+00/))
- type(intergedatadb), parameter **ge17** = interGEdatadb(eosid = "SRK", mruleid = "HV2", ref = "Default/Maribo-Mogensen", bib_ref = "10.1002/aic.14829", uid1 = "CO2", uid2 = "CL-", kijvalue = 0.00000000, correlation = 2, alphaijvalue = (/0.00000000e+00, 0.00000000e+00/), polyij = (/7.24800000e+02, 0.00000000e+00, 1.↩ 00000000e+00/), polyji = (/7.24800000e+02, 0.00000000e+00, 1.00000000e+00/))
- type(intergedatadb), parameter **ge18** = interGEdatadb(eosid = "SRK", mruleid = "HV2", ref = "Default/Maribo-Mogensen", bib_ref = "10.1002/aic.14829", uid1 = "C1", uid2 = "NA+", kijvalue = 0.00000000, correlation = 2, alphaijvalue = (/0.00000000e+00, 0.00000000e+00/), polyij = (/1.12800000e+03, 0.00000000e+00, 1.↩ 00000000e+00/), polyji = (/1.12800000e+03, 0.00000000e+00, 1.00000000e+00/))
- type(intergedatadb), parameter **ge19** = interGEdatadb(eosid = "SRK", mruleid = "HV2", ref = "Default/Maribo-Mogensen", bib_ref = "10.1002/aic.14829", uid1 = "C1", uid2 = "CL-", kijvalue = 0.00000000, correlation = 2, alphaijvalue = (/0.00000000e+00, 0.00000000e+00/), polyij = (/1.12800000e+03, 0.00000000e+00, 1.↩ 00000000e+00/), polyji = (/1.12800000e+03, 0.00000000e+00, 1.00000000e+00/))
- type(lijdatadb), parameter **lij1** = lijdatadb(eosid = "PR", mruleid = "vdW", ref = "QuantumCubic", bib_ref = "10.1016/j.fluid.2020.112790", uid1 = "H2", uid2 = "HE", lijvalue = -0.16000000)
- type(cpakijdata), parameter **cpa1** = CPAkijdata(eosid = "CPA-SRK", ref = "DEFAULT", bib_ref = "", uid1 = "C3", uid2 = "H2O", kij_a = 0.11350000, kij_eps = 0.00000000, kij_beta = 0.00000000, eps_comb_rule = ariComb, beta_comb_rule = geoComb)
- type(cpakijdata), parameter **cpa2** = CPAkijdata(eosid = "CPA-SRK", ref = "DEFAULT", bib_ref = "", uid1 = "NC4", uid2 = "H2O", kij_a = 0.08750000, kij_eps = 0.00000000, kij_beta = 0.00000000, eps_comb_rule = ariComb, beta_comb_rule = geoComb)
- type(cpakijdata), parameter **cpa3** = CPAkijdata(eosid = "CPA-SRK", ref = "DEFAULT", bib_ref = "", uid1 = "NC5", uid2 = "H2O", kij_a = 0.06150000, kij_eps = 0.00000000, kij_beta = 0.00000000, eps_comb_rule = ariComb, beta_comb_rule = geoComb)
- type(cpakijdata), parameter **cpa4** = CPAkijdata(eosid = "CPA-SRK", ref = "DEFAULT", bib_ref = "", uid1 = "NC6", uid2 = "H2O", kij_a = 0.03550000, kij_eps = 0.00000000, kij_beta = 0.00000000, eps_comb_rule = ariComb, beta_comb_rule = geoComb)
- type(cpakijdata), parameter **cpa5** = CPAkijdata(eosid = "CPA-SRK", ref = "DEFAULT", bib_ref = "", uid1 = "NC7", uid2 = "H2O", kij_a = 0.00950000, kij_eps = 0.00000000, kij_beta = 0.00000000, eps_comb_rule = ariComb, beta_comb_rule = geoComb)
- type(cpakijdata), parameter **cpa6** = CPAkijdata(eosid = "CPA-SRK", ref = "DEFAULT", bib_ref = "", uid1 = "NC8", uid2 = "H2O", kij_a = -0.01650000, kij_eps = 0.00000000, kij_beta = 0.00000000, eps_comb_rule = ariComb, beta_comb_rule = geoComb)

- type(cpakijdata), parameter **cpa7** = CPAkijdata(eosid = "CPA-SRK", ref = "DEFAULT", bib_ref = "", uid1 = "NC10", uid2 = "H2O", kij_a = -0.06850000, kij_eps = 0.00000000, kij_beta = 0.00000000, eps_comb_rule = ariComb, beta_comb_rule = geoComb)
- type(cpakijdata), parameter **cpa8** = CPAkijdata(eosid = "CPA-SRK", ref = "DEFAULT", bib_ref = "", uid1 = "C3", uid2 = "MEOH", kij_a = 0.05900000, kij_eps = 0.00000000, kij_beta = 0.00000000, eps_comb_rule = ariComb, beta_comb_rule = geoComb)
- type(cpakijdata), parameter **cpa9** = CPAkijdata(eosid = "CPA-SRK", ref = "DEFAULT", bib_ref = "", uid1 = "MEOH", uid2 = "H2O", kij_a = -0.09000000, kij_eps = 0.00000000, kij_beta = 0.00000000, eps_comb_rule = ariComb, beta_comb_rule = geoComb)
- type(cpakijdata), parameter **cpa10** = CPAkijdata(eosid = "CPA-SRK", ref = "DEFAULT", bib_ref = "", uid1 = "ETOH", uid2 = "H2O", kij_a = -0.11000000, kij_eps = 0.00000000, kij_beta = 0.00000000, eps_comb_rule = ariComb, beta_comb_rule = geoComb)
- type(cpakijdata), parameter **cpa11** = CPAkijdata(eosid = "CPA-SRK", ref = "DEFAULT", bib_ref = "", uid1 = "CO2", uid2 = "H2O", kij_a = 0.04626056, kij_eps = 0.06022255, kij_beta = 0.00000000, eps_comb_rule = ariComb, beta_comb_rule = geoComb)
- type(cpakijdata), parameter **cpa12** = CPAkijdata(eosid = "CPA-PR", ref = "DEFAULT", bib_ref = "", uid1 = "CO2", uid2 = "H2O", kij_a = 0.03000000, kij_eps = 0.00000000, kij_beta = 0.00000000, eps_comb_rule = geoComb, beta_comb_rule = geoComb)
- integer, parameter **maxkij** =723
- type(kijdatadb), dimension(maxkij), parameter **kijdb** = (/ vdw1,vdw2,vdw3,vdw4,vdw5, vdw6,vdw7,vdw8,vdw9,vdw10, vdw11,vdw12,vdw13,vdw14,vdw15, vdw16,vdw17,vdw18,vdw19,vdw20, vdw21,vdw22,vdw23,vdw24,vdw25, vdw26,vdw27,vdw28,vdw29,vdw30, vdw31,vdw32,vdw33,vdw34,vdw35, vdw36,vdw37,vdw38,vdw39,vdw40, vdw41,vdw42,vdw43,vdw44,vdw45, vdw46,vdw47,vdw48,vdw49,vdw50, vdw51,vdw52,vdw53,vdw54,vdw55, vdw56,vdw57,vdw58,vdw59,vdw60, vdw61,vdw62,vdw63,vdw64,vdw65, vdw66,vdw67,vdw68,vdw69,vdw70, vdw71,vdw72,vdw73,vdw74,vdw75, vdw76,vdw77,vdw78,vdw79,vdw80, vdw81,vdw82,vdw83,vdw84,vdw85, vdw86,vdw87,vdw88,vdw89,vdw90, vdw91,vdw92,vdw93,vdw94,vdw95, vdw96,vdw97,vdw98,vdw99,vdw100, vdw101,vdw102,vdw103,vdw104,vdw105, vdw106,vdw107,vdw108,vdw109,vdw110, vdw111,vdw112,vdw113,vdw114,vdw115 vdw116,vdw117,vdw118,vdw119,vdw120, vdw121,vdw122,vdw123,vdw124,vdw125, vdw126,vdw127,vdw128,vdw129,vdw130 vdw131,vdw132,vdw133,vdw134,vdw135, vdw136,vdw137,vdw138,vdw139,vdw140, vdw141,vdw142,vdw143,vdw144,vdw145 vdw146,vdw147,vdw148,vdw149,vdw150, vdw151,vdw152,vdw153,vdw154,vdw155, vdw156,vdw157,vdw158,vdw159,vdw160 vdw161,vdw162,vdw163,vdw164,vdw165, vdw166,vdw167,vdw168,vdw169,vdw170, vdw171,vdw172,vdw173,vdw174,vdw175 vdw176,vdw177,vdw178,vdw179,vdw180, vdw181,vdw182,vdw183,vdw184,vdw185, vdw186,vdw187,vdw188,vdw189,vdw190 vdw191,vdw192,vdw193,vdw194,vdw195, vdw196,vdw197,vdw198,vdw199,vdw200, vdw201,vdw202,vdw203,vdw204,vdw205 vdw206,vdw207,vdw208,vdw209,vdw210, vdw211,vdw212,vdw213,vdw214,vdw215, vdw216,vdw217,vdw218,vdw219,vdw220 vdw221,vdw222,vdw223,vdw224,vdw225, vdw226,vdw227,vdw228,vdw229,vdw230, vdw231,vdw232,vdw233,vdw234,vdw235 vdw236,vdw237,vdw238,vdw239,vdw240, vdw241,vdw242,vdw243,vdw244,vdw245, vdw246,vdw247,vdw248,vdw249,vdw250 vdw251,vdw252,vdw253,vdw254,vdw255, vdw256,vdw257,vdw258,vdw259,vdw260, vdw261,vdw262,vdw263,vdw264,vdw265 vdw266,vdw267,vdw268,vdw269,vdw270, vdw271,vdw272,vdw273,vdw274,vdw275, vdw276,vdw277,vdw278,vdw279,vdw280 vdw281,vdw282,vdw283,vdw284,vdw285, vdw286,vdw287,vdw288,vdw289,vdw290, vdw291,vdw292,vdw293,vdw294,vdw295 vdw296,vdw297,vdw298,vdw299,vdw300, vdw301,vdw302,vdw303,vdw304,vdw305, vdw306,vdw307,vdw308,vdw309,vdw310 vdw311,vdw312,vdw313,vdw314,vdw315, vdw316,vdw317,vdw318,vdw319,vdw320, vdw321,vdw322,vdw323,vdw324,vdw325 vdw326,vdw327,vdw328,vdw329,vdw330, vdw331,vdw332,vdw333,vdw334,vdw335, vdw336,vdw337,vdw338,vdw339,vdw340 vdw341,vdw342,vdw343,vdw344,vdw345, vdw346,vdw347,vdw348,vdw349,vdw350, vdw351,vdw352,vdw353,vdw354,vdw355 vdw356,vdw357,vdw358,vdw359,vdw360, vdw361,vdw362,vdw363,vdw364,vdw365, vdw366,vdw367,vdw368,vdw369,vdw370 vdw371,vdw372,vdw373,vdw374,vdw375, vdw376,vdw377,vdw378,vdw379,vdw380, vdw381,vdw382,vdw383,vdw384,vdw385 vdw386,vdw387,vdw388,vdw389,vdw390, vdw391,vdw392,vdw393,vdw394,vdw395, vdw396,vdw397,vdw398,vdw399,vdw400 vdw401,vdw402,vdw403,vdw404,vdw405, vdw406,vdw407,vdw408,vdw409,vdw410, vdw411,vdw412,vdw413,vdw414,vdw415 vdw416,vdw417,vdw418,vdw419,vdw420, vdw421,vdw422,vdw423,vdw424,vdw425, vdw426,vdw427,vdw428,vdw429,vdw430 vdw431,vdw432,vdw433,vdw434,vdw435, vdw436,vdw437,vdw438,vdw439,vdw440, vdw441,vdw442,vdw443,vdw444,vdw445 vdw446,vdw447,vdw448,vdw449,vdw450, vdw451,vdw452,vdw453,vdw454,vdw455, vdw456,vdw457,vdw458,vdw459,vdw460 vdw461,vdw462,vdw463,vdw464,vdw465, vdw466,vdw467,vdw468,vdw469,vdw470, vdw471,vdw472,vdw473,vdw474,vdw475 vdw476,vdw477,vdw478,vdw479,vdw480, vdw481,vdw482,vdw483,vdw484,vdw485, vdw486,vdw487,vdw488,vdw489,vdw490 vdw491,vdw492,vdw493,vdw494,vdw495, vdw496,vdw497,vdw498,vdw499,vdw500, vdw501,vdw502,vdw503,vdw504,vdw505 vdw506,vdw507,vdw508,vdw509,vdw510, vdw511,vdw512,vdw513,vdw514,vdw515, vdw516,vdw517,vdw518,vdw519,vdw520 vdw521,vdw522,vdw523,vdw524,vdw525, vdw526,vdw527,vdw528,vdw529,vdw530, vdw531,vdw532,vdw533,vdw534,vdw535 vdw536,vdw537,vdw538,vdw539,vdw540, vdw541,vdw542,vdw543,vdw544,vdw545, vdw546,vdw547,vdw548,vdw549,vdw550 vdw551,vdw552,vdw553,vdw554,vdw555, vdw556,vdw557,vdw558,vdw559,vdw560, vdw561,vdw562,vdw563,vdw564,vdw565 vdw566,vdw567,vdw568,vdw569,vdw570, vdw571,vdw572,vdw573,vdw574,vdw575, vdw576,vdw577,vdw578,vdw579,vdw580

vdw581,vdw582,vdw583,vdw584,vdw585, vdw586,vdw587,vdw588,vdw589,vdw590, vdw591,vdw592,vdw593,vdw594,vdw595
vdw596,vdw597,vdw598,vdw599,vdw600, vdw601,vdw602,vdw603,vdw604,vdw605, vdw606,vdw607,vdw608,vdw609,vdw610
vdw611,vdw612,vdw613,vdw614,vdw615, vdw616,vdw617,vdw618,vdw619,vdw620, vdw621,vdw622,vdw623,vdw624,vdw625
vdw626,vdw627,vdw628,vdw629,vdw630, vdw631,vdw632,vdw633,vdw634,vdw635, vdw636,vdw637,vdw638,vdw639,vdw640
vdw641,vdw642,vdw643,vdw644,vdw645, vdw646,vdw647,vdw648,vdw649,vdw650, vdw651,vdw652,vdw653,vdw654,vdw655
vdw656,vdw657,vdw658,vdw659,vdw660, vdw661,vdw662,vdw663,vdw664,vdw665, vdw666,vdw667,vdw668,vdw669,vdw670
vdw671,vdw672,vdw673,vdw674,vdw675, vdw676,vdw677,vdw678,vdw679,vdw680, vdw681,vdw682,vdw683,vdw684,vdw685
vdw686,vdw687,vdw688,vdw689,vdw690, vdw691,vdw692,vdw693,vdw694,vdw695, vdw696,vdw697,vdw698,vdw699,vdw700
vdw701,vdw702,vdw703,vdw704,vdw705, vdw706,vdw707,vdw708,vdw709,vdw710, vdw711,vdw712,vdw713,vdw714,vdw715
vdw716,vdw717,vdw718,vdw719,vdw720, vdw721,vdw722,vdw723 /)

- integer, parameter **maxintergeij** =19
- type(intergedatadb), dimension(maxintergeij), parameter **intergedb** = (/ ge1,ge2,ge3,ge4,ge5, ge6,ge7,ge8,ge9,ge10, ge11,ge12,ge13,ge14,ge15, ge16,ge17,ge18,ge19 /)
- integer, parameter **maxlij** =1
- type(lijdatadb), dimension(maxlij), parameter **lijdb** = (/ lij1 /)
- integer, parameter **cpamaxkij** =12
- type(cpakijdata), dimension(cpamaxkij), parameter **cpakijdb** = (/ cpa1,cpa2,cpa3,cpa4,cpa5, cpa6,cpa7,cpa8,cpa9,cpa10, cpa11,cpa12 /)

5.34.1 Detailed Description

Automatically generated to file mixdatadb.f90 using utility python code pyUtils Time stamp: 2023-02-27T15:11↩ :46.339247.

5.35 multiparameter_base Module Reference

Definitions of adimensional variables: delta = rho/rho_c tau = T_c/T alpha0 = A^{ideal}/(nRT) alphaRes = A^{res}/(nRT) alpha = A/(nRT)

Data Types

- interface alpha0_hd_intf
- interface alpha0derivs_intf
- interface alphares_hd_intf
- interface alpharesderivs_intf
- interface assign_meos_intf
- interface init_intf
- type meos

  Base class for multiparameter equations of state.
- type nist_meos_ptr
- interface satdeltaestimate_intf

Functions/Subroutines

- real function cv (this, t, v)

  Isochoric heat capacity.
- real function cp (this, t, v)

  Isobaric heat capacity.
- real function speed_of_sound (this, t, v)

  Speed of sound.
- subroutine **getcritpoint** (this, tcrit, pcrit, rhocrit)
- subroutine **calc_zfac** (this, t, p, n, phase, z, z_t, z_p, z_n)
- subroutine **calc_lnphi** (this, t, p, n, phase, lnphi, lnphi_t, lnphi_p, lnphi_n)
- subroutine **calc_entropy** (this, t, p, n, phase, s, s_t, s_p, s_n, residual)
- subroutine **calc_enthalpy** (this, t, p, n, phase, h, h_t, h_p, h_n, residual)

- subroutine **alpha_to_f_conversion** (this, t, v, n, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- subroutine **calc_f** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- subroutine **calc_fid** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)

  Calculate reduced ideal Helmholtz energy and differentials.
- subroutine **calc_resgibbs** (this, t, p, n, phase, g, g_t, g_p, g_n)
- subroutine alpharesderivs_tv (this, t, v, alpr, alpr_t, alpr_v, alpr_tt, alpr_tv, alpr_vv, alpr_n, alpr_tn, alpr_vn, alpr_nn)
- subroutine alphaderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, residual)
- subroutine alphaidderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, alp_n, alp_tn, alp_vn, alp_nn)
- subroutine densitysolver (this, t_spec, p_spec, phase_spec, rho, phase_found, ierr)
- subroutine **mp_pressure** (this, rho, t, p, p_rho, p_t)

  Pressure and (optionally) its derivatives.
- subroutine **assign_meos_base** (this, other)
- subroutine **get_ref_state_spec_default** (this, ref_state, t, p, phase, solve)
- subroutine **set_ref_state_default** (this, t, p, v, h, s)

Variables

- real, parameter **releps_p** = machine_prec∗1e8
- real, parameter **releps_rho** = machine_prec∗1e6
- integer, parameter **ref_no_solve** =1
- integer, parameter **ref_evaluate_id** =2
- integer, parameter **ref_solve_for_t** =3
- integer, parameter **ref_solve_for_p** =4

5.35.1 Detailed Description

Definitions of adimensional variables: delta = rho/rho_c tau = T_c/T alpha0 = A^{ideal}/(nRT) alphaRes = A^{res}/(nRT) alpha = A/(nRT)

5.35.2 Function/Subroutine Documentation

5.35.2.1 alphaderivs_tv()

```
subroutine multiparameter_base::alphaderivs_tv (
            class(meos) this,
            real, intent(in) t,
            real, intent(in) v,
            real, intent(out), optional alp,
            real, intent(out), optional alp_t,
            real, intent(out), optional alp_v,
            real, intent(out), optional alp_tt,
            real, intent(out), optional alp_tv,
            real, intent(out), optional alp_vv,
            logical, intent(in), optional residual )
```

Parameters

	this	Calling class
in	*v*	Temperature (K) and molar volume (m^3/mol)
out	*alp*	A/(nRT)

5.35.2.2 alphaidderivs_tv()

```
subroutine multiparameter_base::alphaidderivs_tv (
            class(meos) this,
            real, intent(in) t,
            real, intent(in) v,
            real, intent(out), optional alp,
            real, intent(out), optional alp_t,
            real, intent(out), optional alp_v,
            real, intent(out), optional alp_tt,
            real, intent(out), optional alp_tv,
            real, intent(out), optional alp_vv,
            real, intent(out), optional alp_n,
            real, intent(out), optional alp_tn,
            real, intent(out), optional alp_vn,
            real, intent(out), optional alp_nn )
```

Parameters

	this	Calling class
in	*v*	Temperature (K) and molar volume (m^3/mol)
out	*alp*	A/(nRT)

5.35.2.3 alpharesderivs_tv()

```
subroutine multiparameter_base::alpharesderivs_tv (
            class(meos) this,
            real, intent(in) t,
            real, intent(in) v,
            real, intent(out), optional alpr,
            real, intent(out), optional alpr_t,
            real, intent(out), optional alpr_v,
            real, intent(out), optional alpr_tt,
            real, intent(out), optional alpr_tv,
            real, intent(out), optional alpr_vv,
            real, intent(out), optional alpr_n,
            real, intent(out), optional alpr_tn,
            real, intent(out), optional alpr_vn,
            real, intent(out), optional alpr_nn )
```

Parameters

	this	Calling class
in	*v*	Temperature (K) and molar volume (m^3/mol)
out	*alpr*	A/(nRT)

5.35.2.4 cp()

```
real function multiparameter_base::cp (
            class(meos) this,
            real, intent(in) t,
            real, intent(in) v )
```

Isobaric heat capacity.

Parameters

	this	Calling class
in	*v*	Temperature (K) and molar volume (m^3/mol)

Returns

[J/(mol∗K)]

5.35.2.5 cv()

```
real function multiparameter_base::cv (
             class(meos) this,
             real, intent(in) t,
             real, intent(in) v )
```

Isochoric heat capacity.

Parameters

	this	Calling class
in	*v*	Temperature (K) and molar volume (m^3/mol)

Returns

[J/(mol∗K)]

5.35.2.6 densitysolver()

```
subroutine multiparameter_base::densitysolver (
             class(meos) this,
             real, intent(in) t_spec,
             real, intent(in) p_spec,
             integer, intent(in) phase_spec,
             real, intent(out) rho,
             integer, intent(out), optional phase_found,
             integer, intent(out), optional ierr )
```

Parameters

	this	The calling class.
in	*p_spec*	Temperature (K) and pressure (Pa)
in	*phase_spec*	Phase flag.
out	*rho*	Density (mol/m^3)

5.35.2.7 speed_of_sound()

```
real function multiparameter_base::speed_of_sound (
             class(meos) this,
             real, intent(in) t,
             real, intent(in) v )
```

Speed of sound.

Parameters

	this	Calling class

Parameters

in	*v*	Temperature (K) and molar volume (m^3/mol)

Returns

[m/s]

5.36 multiparameter_idealmix Module Reference

This module mixes multiparameter EoS through an ideal mixing rule. It relies heavily on the multiparameter EoS framwork and the methodology implemented by Ailo Aasen. 2018-10-01 Oivind Wilhelmsen.

Functions/Subroutines

- subroutine, public calc_multiparameter_idealmix_zfac (nc, meos, t, p, z, phase, zfac, dzdt, dzdp, dzdz)

  Initialize framework for computing thermodynamic properties with an ideal mixture of multiparameter EoS. NB: Only expected to give reasonable results in the single-phase regions, two-phase regions should be avoided due to possible multiple maxwell loops ++.
- subroutine, public calc_multiparameter_idealmix_enthalpy (nc, meos, t, p, z, phase, h, dhdt, dhdp, dhdz)
- subroutine, public calc_multiparameter_idealmix_entropy (nc, meos, t, p, z, phase, s, dsdt, dsdp, dsdz)

  This function calculates the residual Entropy and the derivatives.
- subroutine, public calc_multiparameter_idealmix_fugacity (nc, meos, t, p, z, phase, fug, dlnfdt, dlnfdp, dlnfdz)

  This function calculates the Fugacity coefficient and the derivatives.
- subroutine, public calc_multiparameter_idealmix_gres (nc, meos, t, p, z, phase, gr, dgrdt, dgrdp)

  This subroutine calculates the residual gibbs energy and the derivatives.

5.36.1 Detailed Description

This module mixes multiparameter EoS through an ideal mixing rule. It relies heavily on the multiparameter EoS framwork and the methodology implemented by Ailo Aasen. 2018-10-01 Oivind Wilhelmsen.

5.36.2 Function/Subroutine Documentation

5.36.2.1 calc_multiparameter_idealmix_enthalpy()

```
subroutine, public multiparameter_idealmix::calc_multiparameter_idealmix_enthalpy (
             integer, intent(in) nc,
             class(meos_idealmix), intent(in) meos,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             integer, intent(in) phase,
             real, intent(out) h,
             real, intent(out), optional dhdt,
             real, intent(out), optional dhdp,
             real, dimension(nc), intent(out), optional dhdz )
```

This function calculates the residual Enthalpy and the derivatives.

Parameters

T	The temperature [K]
P	The pressure [Pa]
Z	The mole fraction [-]
phase	The phase, 1=liquid, 2=vapour
enthalpy	h [J/mol]

Parameters

dhdt	Temperature derivative [J/mole K]
dhdp	Pressure derivative [J/mole Pa]
dhdz	Composition derivative [J/mole^2]

5.36.2.2 Author: OW, date: 2018-10-02

Set to zero
We extract only the residual values (Excluding ideal gas terms)
Check if the optionals are present

5.36.2.3 calc_multiparameter_idealmix_entropy()

```
subroutine, public multiparameter_idealmix::calc_multiparameter_idealmix_entropy (
              integer, intent(in) nc,
              class(meos_idealmix), intent(in) meos,
              real, intent(in) t,
              real, intent(in) p,
              real, dimension(nc), intent(in) z,
              integer, intent(in) phase,
              real, intent(out) s,
              real, intent(out), optional dsdt,
              real, intent(out), optional dsdp,
              real, dimension(nc), intent(out), optional dsdz )
```

This function calculates the residual Entropy and the derivatives.

Parameters

T	The temperature [K]
P	The pressure [Pa]
Z	The overall mole fraction [-]
phase	The phase, 1=liquid, 2=vapour
dsdt	Temperature derivative [J/kmoleK]
dsdp	Pressure derivative [J/kmolePa]
dsdz	Composition derivative [J/kmole^2]
numder	Analytical derivatives if true (default false)

Return values

entropy	The entropy [J/kmole K]

5.36.2.4 Author: OW, date: 2018-10-09

Set to zero
We extract only the residual values (Excluding ideal gas terms)
Check if the optionals are present
! NB: S_mix=sum_i^N N_iS_i

5.36.2.5 calc_multiparameter_idealmix_fugacity()

```
subroutine, public multiparameter_idealmix::calc_multiparameter_idealmix_fugacity (
              integer, intent(in) nc,
```

```
            class(meos_idealmix), intent(in) meos,
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) z,
            integer, intent(in) phase,
            real, dimension(nc), intent(out) fug,
            real, dimension(nc), intent(out), optional dlnfdt,
            real, dimension(nc), intent(out), optional dlnfdp,
            real, dimension(nc,nc), intent(out), optional dlnfdz )
```

This function calculates the Fugacity coefficient and the derivatives.

Parameters

T	The temperature [K]
P	The pressure [Pa]
Z	The overall mole fraction [-]
phase	The phase, 1=liquid, 2=vapour
fug,Fugacity	coefficient []
dlnfdt	Temperature derivative [1/K] (dlog(f)/dT)
dlnfdp	Pressure derivative [1/Pa] (dlog(f)/dP)
dlnfdz	Composition derivative [1/kmole] (dlog(f)/dNi)

5.36.2.6 Author: OW, date: 2018-10-09

Check if the optionals are present, if they are, call subroutines with optionals

5.36.2.7 calc_multiparameter_idealmix_gres()

```
subroutine, public multiparameter_idealmix::calc_multiparameter_idealmix_gres (
            integer, intent(in) nc,
            class(meos_idealmix), intent(in) meos,
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) z,
            integer, intent(in) phase,
            real, intent(out) gr,
            real, intent(out), optional dgrdt,
            real, intent(out), optional dgrdp )
```

This subroutine calculates the residual gibbs energy and the derivatives.

Parameters

T	The temperature [K]
P	The pressure [Pa]
Z	The overall mole fraction [-]
phase	The phase, 1=liquid, 2=vapour
gr	Residual gibbs energy [J/kmol]
dgrdt	Temperature derivative [J/kmol/K]
dgrdp	Pressure derivative [J/kmol/Pa]

5.36.2.8 Author: OW, date: 2018-10-09

Check if the optionals are present

5.36.2.9 calc_multiparameter_idealmix_zfac()

```
subroutine, public multiparameter_idealmix::calc_multiparameter_idealmix_zfac (
             integer, intent(in) nc,
             class(meos_idealmix), intent(in) meos,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             integer, intent(in) phase,
             real, intent(out) zfac,
             real, intent(out), optional dzdt,
             real, intent(out), optional dzdp,
             real, dimension(nc), intent(out), optional dzdz )
```

Initialize framework for computing thermodynamic properties with an ideal mixture of multiparameter EoS. NB↩ : Only expected to give reasonable results in the single-phase regions, two-phase regions should be avoided due to possible multiple maxwell loops ++.
\AUTHOR OW, date:2018-10-01 This function calculates the Compressibility factor with derivatives.

Parameters

T	The temperature [K]
P	The pressure [Pa]
Z	The mole fraction [-]
phase	The phase, 1=liquid, 2=vapour
Zfac	The Z-factor [-]
dZdt	Temperature derivative [1/K]
dZdp	Pressure derivative [1/Pa]
dZdz	Composition derivative [-]

5.36.2.10 Author: OW, date: 2018-10-02

Set to zero
Check if the optionals are present

5.37 multipol Module Reference

Residual reduced Helmholtz energies from dipol amd quadrupol interactions: Gross 2005: 10.1002/aic.10502 Gross and Vrabec 2006: 10.1002/aic.10683 Vrabec and Gross 2008: 10.1021/jp072619u.

Functions/Subroutines

- type(hyperdual) function, dimension(nce) hyperdual_calc_d_hs_pc_saft (eos, nce, t)

  Calculate the PC-SAFT hard-sphere segment diameter d_i.
- type(hyperdual) function hyperdual_packing_fraction_pc_saft (eos, nce, v, n, d_hs)

  Calculate the PC-SAFT packing fraction using the hard-sphere diameter.
- type(hyperdual) function, public hyperdual_fres_multipol (p_eos, nc, t, v, n)

  Calculate reduced helmholtz energy from quadrupoles and dipoles.
- type(hyperdual) function hyperdual_j2_ij (nce, t, eta, a_ij, b_ij, i, j, lmax)

  Calculate correlation integral J2.
- type(hyperdual) function hyperdual_j3_ijk (nce, eta, c_ijk, i, j, k, lmax)

  Calculate correlation integral J3.
- type(hyperdual) function hyperdual_f_qq (nce, t, v, n, eta, mpol_param)

  Quadrupol-quadrupol interaction Gross 2005: 10.1002/aic.10502.
- type(hyperdual) function hyperdual_f_dd (nce, t, v, n, eta, mpol_param)

Dipol-dipol interaction Gross and Vrabec 2006: 10.1002/aic.10683.

- type(hyperdual) function hyperdual_f_dq (nce, t, v, n, eta, mpol_param)

  Quadrupol-dipol interaction Vrabec and Gross 2008: 10.1021/jp072619u.
- subroutine, public add_hyperdual_fres_multipol (eos, nc, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_vv, f_tn, f_vn, f_nn)

  Real variable wrapper around hyperdual numbers. Calculate reduced Helmholtz energy and differentials,.
- subroutine, public fres_multipol (t, v, n, qq, dd, dq, f)

  Test function for polar reduced Helmholtz energy.
- subroutine, public multipol_model_control (qq, dd, dq)

  Enable/disable polar contributions.

Variables

- real, parameter **alpha** = 1.1937350

5.37.1 Detailed Description

Residual reduced Helmholtz energies from dipol amd quadrupol interactions: Gross 2005: 10.1002/aic.10502 Gross and Vrabec 2006: 10.1002/aic.10683 Vrabec and Gross 2008: 10.1021/jp072619u.

5.37.2 Function/Subroutine Documentation

5.37.2.1 add_hyperdual_fres_multipol()

```
subroutine, public multipol::add_hyperdual_fres_multipol (
             class(base_eos_param), intent(inout) eos,
             integer, intent(in) nc,
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(nc), intent(in) n,
             real, intent(out), optional f,
             real, intent(out), optional f_t,
             real, intent(out), optional f_v,
             real, dimension(nc), intent(out), optional f_n,
             real, intent(out), optional f_tt,
             real, intent(out), optional f_tv,
             real, intent(out), optional f_vv,
             real, dimension(nc), intent(out), optional f_tn,
             real, dimension(nc), intent(out), optional f_vn,
             real, dimension(nc,nc), intent(out), optional f_nn )
```

Real variable wrapper around hyperdual numbers. Calculate reduced Helmholtz energy and differentials,.

Author

Morten Hammer, 2022

5.37.2.2 fres_multipol()

```
subroutine, public multipol::fres_multipol (
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(nce), intent(in) n,
             logical, intent(in) qq,
             logical, intent(in) dd,
             logical, intent(in) dq,
             real, intent(out) f )
```

Test function for polar reduced Helmholtz energy.

Author

Morten Hammer, 2023-02

5.37.2.3 hyperdual_calc_d_hs_pc_saft()

```
type(hyperdual) function, dimension(nce) multipol::hyperdual_calc_d_hs_pc_saft (
             class(spcsaft_eos), intent(in) eos,
             integer, intent(in) nce,
             type(hyperdual), intent(in) t )
```

Calculate the PC-SAFT hard-sphere segment diameter d_i.

Author

Morten Hammer, 2022

Parameters

in	*t*	[K]

Returns

[m]

5.37.2.4 hyperdual_f_dd()

```
type(hyperdual) function multipol::hyperdual_f_dd (
             integer, intent(in) nce,
             type(hyperdual), intent(in) t,
             type(hyperdual), intent(in) v,
             type(hyperdual), dimension(nce), intent(in) n,
             type(hyperdual), intent(in) eta,
             class(multipol_param), intent(in) mpol_param )
```

Dipol-dipol interaction Gross and Vrabec 2006: 10.1002/aic.10683.

Author

Morten Hammer, 2022

5.37.2.5 hyperdual_f_dq()

```
type(hyperdual) function multipol::hyperdual_f_dq (
             integer, intent(in) nce,
             type(hyperdual), intent(in) t,
             type(hyperdual), intent(in) v,
             type(hyperdual), dimension(nce), intent(in) n,
             type(hyperdual), intent(in) eta,
             class(multipol_param), intent(in) mpol_param )
```

Quadrupol-dipol interaction Vrabec and Gross 2008: 10.1021/jp072619u.

Author

Morten Hammer, 2022

5.37.2.6 hyperdual_f_qq()

```
type(hyperdual) function multipol::hyperdual_f_qq (
             integer, intent(in) nce,
```

```
            type(hyperdual), intent(in) t,
            type(hyperdual), intent(in) v,
            type(hyperdual), dimension(nce), intent(in) n,
            type(hyperdual), intent(in) eta,
            class(multipol_param), intent(in) mpol_param )
```

Quadrupol-quadrupol interaction Gross 2005: 10.1002/aic.10502.

Author

Morten Hammer, 2022

5.37.2.7 hyperdual_fres_multipol()

```
type(hyperdual) function, public multipol::hyperdual_fres_multipol (
            class(base_eos_param), intent(inout) p_eos,
            integer, intent(in) nc,
            type(hyperdual), intent(in) t,
            type(hyperdual), intent(in) v,
            type(hyperdual), dimension(nc), intent(in) n )
```

Calculate reduced helmholtz energy from quadrupoles and dipoles.

Author

Morten Hammer, 2022

5.37.2.8 hyperdual_j2_ij()

```
type(hyperdual) function multipol::hyperdual_j2_ij (
            integer, intent(in) nce,
            type(hyperdual), intent(in) t,
            type(hyperdual), intent(in) eta,
            real, dimension(0:lmax,nce,nce), intent(in) a_ij,
            real, dimension(0:lmax,nce,nce), intent(in) b_ij,
            integer, intent(in) i,
            integer, intent(in) j,
            integer, intent(in) lmax )
```

Calculate correlation integral J2.

Author

Morten Hammer, 2022

5.37.2.9 hyperdual_j3_ijk()

```
type(hyperdual) function multipol::hyperdual_j3_ijk (
            integer, intent(in) nce,
            type(hyperdual), intent(in) eta,
            real, dimension(0:lmax,nce,nce,nce), intent(in) c_ijk,
            integer, intent(in) i,
            integer, intent(in) j,
            integer, intent(in) k,
            integer, intent(in) lmax )
```

Calculate correlation integral J3.

Author

Morten Hammer, 2022

5.37.2.10 hyperdual_packing_fraction_pc_saft()

```
type(hyperdual) function multipol::hyperdual_packing_fraction_pc_saft (
             class(spcsaft_eos), intent(in) eos,
             integer, intent(in) nce,
             type(hyperdual), intent(in) v,
             type(hyperdual), dimension(nce), intent(in) n,
             type(hyperdual), dimension(nce), intent(in) d_hs )
```

Calculate the PC-SAFT packing fraction using the hard-sphere diameter.

Author

Morten Hammer, 2022

5.37.2.11 multipol_model_control()

```
subroutine, public multipol::multipol_model_control (
             logical, intent(in) qq,
             logical, intent(in) dd,
             logical, intent(in) dq )
```

Enable/disable polar contributions.

Author

Morten Hammer, 2023-02

5.38 mut_solver Module Reference

Solve mu-T problem. Look for single phase solution given initial guess.

Functions/Subroutines

- subroutine, public solve_mu_t (mu, t, rho, ierr)

  Solve mu(T,rho) = mu for a given temperature.
- subroutine, public solve_lnf_t (lnf, t, rho, ierr)

  Solve lnf(T,rho) = lnf for a given temperature.
- subroutine, public map_meta_isotherm (t, z, n, phase, v, rho_other, ierr)

  Map isotherm from saturation line to spinodal.

5.38.1 Detailed Description

Solve mu-T problem. Look for single phase solution given initial guess.

5.38.2 Function/Subroutine Documentation

5.38.2.1 map_meta_isotherm()

```
subroutine, public mut_solver::map_meta_isotherm (
             real, intent(in) t,
             real, dimension(nce), intent(in) z,
             integer, intent(in) n,
             integer, intent(in) phase,
             real, dimension(n), intent(out) v,
             real, dimension(n,nce), intent(out) rho_other,
             integer, intent(out) ierr )
```

Map isotherm from saturation line to spinodal.

Author

MH, 2022-11

Parameters

in	*z*	Composition
in	*t*	Temperature [K]
in	*n*	Number of points on isotherm
in	*phase*	Phase identifer
out	*ierr*	Error flag
out	*v*	Specific volume (m3/mol)
out	*rho_other*	Specific volume of other phase (-)

5.38.2.2 solve_lnf_t()

```
subroutine, public mut_solver::solve_lnf_t (
             real, dimension(nce), intent(in) lnf,
             real, intent(in) t,
             real, dimension(nce), intent(inout) rho,
             integer, intent(out) ierr )
```

Solve lnf(T,rho) = lnf for a given temperature.

Author

MH, July 2022

Parameters

in	*lnf*	Fugacity coefficient
in	*t*	K - Temperature
in,out	*rho*	mol/m3 - Specific mole numbers
out	*ierr*	Error flag

5.38.2.3 solve_mu_t()

```
subroutine, public mut_solver::solve_mu_t (
             real, dimension(nce), intent(in) mu,
             real, intent(in) t,
             real, dimension(nce), intent(inout) rho,
             integer, intent(out) ierr )
```

Solve mu(T,rho) = mu for a given temperature.

Author

MH, July 2022

Parameters

in	*mu*	Chemical potential
in	*t*	K - Temperature
in,out	*rho*	mol/m3 - Specific mole numbers
out	*ierr*	Error flag

5.39 nonlinear_solvers Module Reference

This module contains generic methods for solving systems of non-linear equations.

Data Types

- interface function_template
- interface jacobian_template
- type nonlinear_solver

  A type that contains solver information.

Functions/Subroutines

- subroutine, public nonlinear_solve (solver, fun, jac, hess, limit, premterm, setxvar, x, xmin, xmax, param, dim)

  Interface for solver.
- subroutine, public approximate_jacobian (fun, jac, x0, n, param, xmax, xmin)

  Approximate Jacobian numerically using forward differences.
- subroutine, public approximate_jacobian_2nd (fun, jac, x0, n, param, xmax, xmin)

  Approximate Jacobian numerically using central difference. 2nd order.
- subroutine, public approximate_jacobian_4th (fun, jac, x0, n, param, xmax, xmin)

  Approximate Jacobian numerically using central difference. 4th order.
- subroutine, public limit_dx (n, x, xmin, xmax, dx, np, param)

  Limit change in x, dx, to keep x between extreme values: xmin <= x <= xmax.
- logical function, public premreturn (x, dx, param, n, np)

  No premature return.
- logical function, public premterm_at_dx_zero (var, dvar, param, n, np)

  Terminate search when step size is zero.
- subroutine, public setxv (n, nparam, x, dx, xmin, xmax, param, alpha)

  Set variables.
- subroutine, public bracketing_solver (xmin, xmax, fun, x, solver, param)

  Interface for bracketing methods Solve: f(x)=0 for xlow <= x <= xhigh.
- subroutine, public pegasus (xlow, xhigh, fun, x, solver, param)

  A method of Regula Falsi type for finding a simple root of a non-linear equation. Solve: f(x)=0 for xlow <= x <= xhigh REF: AN IMPROVED PEGASUS METHOD FOR ROOT FINDING RICHARD F. KING BIT Numerical Mathematics, 13 (1973), 423-427.
- subroutine, public **ridders_method** (func, param, xmin, xmax, solver, x)

  Ridders' method for root finding.
- subroutine, public **newton_secondorder_singlevar** (func, xinit, xmin, xmax, solver, x, param)

  Second order Newton solver (cubic convergence).
- real function, public newton_1d (fun, x0, param, xmin, xmax)

  A clean, minimal implemention of Newton's method for solving fun(x) = 0 in 1D. Quadratic convergence.

Variables

- integer, parameter, public ns_newton = 1

  Solver aliases.
- integer, parameter, public **ns_succsub** = 2
- integer, parameter, public **ns_newton_ls** = 3

  Include line search in NR iteration.
- integer, parameter, public **ns_pegasus** = 4

  Bracketing solver aliases.
- integer, parameter, public **ns_ridders** = 5

5.39.1 Detailed Description

This module contains generic methods for solving systems of non-linear equations.

Author

MH, August 2012

5.39.2 Function/Subroutine Documentation

5.39.2.1 approximate_jacobian()

```
subroutine, public nonlinear_solvers::approximate_jacobian (
             external fun,
             real, dimension(n,n), intent(out) jac,
             real, dimension(n), intent(in) x0,
             integer, intent(in) n,
             real, dimension(:), intent(inout) param,
             real, dimension(n), intent(in), optional xmax,
             real, dimension(n), intent(in), optional xmin )
```

Approximate Jacobian numerically using forward differences.

Author

KEGT

Parameters

in,out	*param*	Paramaters

5.39.2.2 approximate_jacobian_2nd()

```
subroutine, public nonlinear_solvers::approximate_jacobian_2nd (
             external fun,
             real, dimension(n,n), intent(out) jac,
             real, dimension(n), intent(in) x0,
             integer, intent(in) n,
             real, dimension(:), intent(in) param,
             real, dimension(n), intent(in), optional xmax,
             real, dimension(n), intent(in), optional xmin )
```

Approximate Jacobian numerically using central difference. 2nd order.

Author

GL

Parameters

in	*param*	Paramaters

5.39.2.3 approximate_jacobian_4th()

```
subroutine, public nonlinear_solvers::approximate_jacobian_4th (
             external fun,
             real, dimension(n,n), intent(out) jac,
             real, dimension(n), intent(in) x0,
             integer, intent(in) n,
             real, dimension(:), intent(in) param,
```

```
            real, dimension(n), intent(in), optional xmax,
            real, dimension(n), intent(in), optional xmin )
```

Approximate Jacobian numerically using central difference. 4th order.

Author

GL

Parameters

in	*param*	Paramaters

5.39.2.4 bracketing_solver()

```
subroutine, public nonlinear_solvers::bracketing_solver (
            real, intent(in) xmin,
            real, intent(in) xmax,
            real, external fun,
            real, intent(out) x,
            type(nonlinear_solver), intent(inout) solver,
            real, dimension(:), intent(inout), optional param )
```

Interface for bracketing methods Solve: f(x)=0 for xlow <= x <= xhigh.

Author

MH, Januar 2014

5.39.2.5 limit_dx()

```
subroutine, public nonlinear_solvers::limit_dx (
            integer, intent(in) n,
            real, dimension(n), intent(in) x,
            real, dimension(n), intent(in) xmin,
            real, dimension(n), intent(in) xmax,
            real, dimension(n), intent(inout) dx,
            integer, intent(in) np,
            real, dimension(np), intent(inout) param )
```

Limit change in x, dx, to keep x between extreme values: xmin <= x <= xmax.

Author

MH, August 2012

5.39.2.6 newton_1d()

```
real function, public nonlinear_solvers::newton_1d (
            external subroutine(real, intent(in) x, real, dimension(:), intent(in) param,
real, intent(out) f, real, intent(out), optional df) fun,
            real, intent(in) x0,
            real, dimension(:), intent(inout) param,
            real, intent(in), optional xmin,
            real, intent(in), optional xmax )
```

A clean, minimal implemention of Newton's method for solving fun(x) = 0 in 1D. Quadratic convergence.

Author

Ailo A, December 2014

Parameters

in	*x0*	initial guess
in,out	*param*	additional parameters for fun
in	*xmax*	limits

Returns

the "root"

5.39.2.7 nonlinear_solve()

```
subroutine, public nonlinear_solvers::nonlinear_solve (
             type(nonlinear_solver), intent(inout) solver,
             external fun,
             external jac,
             external hess,
             external limit,
             logical, external premterm,
             external setxvar,
             real, dimension(:), intent(inout) x,
             real, dimension(:), intent(in) xmin,
             real, dimension(:), intent(in) xmax,
             real, dimension(:), intent(inout) param,
             integer, intent(in), optional dim )
```

Interface for solver.

Author

MH, August 2012

Parameters

in,out	*solver*	Solver information
	fun	The non-linear function F(x)=0
	jac	Function to evaluate the Jacobian of F. This can be approximate_jacobian, in which the jacobian is evaluated approximately
	hess	Function to evaluate the Hessian of F. If analyt_hess=false, then this will be found by inverting J.
	limit	Limit step
	premterm	Test for premature termination
	setxvar	Set variables
in,out	*x*	On entry, initial condition. On exit, solution
in	*xmin*	Minimum and maximum limit for x
in	*xmax*	Minimum and maximum limit for x
in,out	*param*	Paramaters
in	*dim*	Problem size

5.39.2.8 pegasus()

```
subroutine, public nonlinear_solvers::pegasus (
             real, intent(in) xlow,
             real, intent(in) xhigh,
             real, external fun,
```

```
            real, intent(out) x,
            type(nonlinear_solver), intent(inout) solver,
            real, dimension(:), intent(inout), optional param )
```

A method of Regula Falsi type for finding a simple root of a non-linear equation. Solve: f(x)=0 for xlow <= x <= xhigh REF: AN IMPROVED PEGASUS METHOD FOR ROOT FINDING RICHARD F. KING BIT Numerical Mathematics, 13 (1973), 423-427.

Author

MH, August 2012

5.39.2.9 premreturn()

```
logical function, public nonlinear_solvers::premreturn (
            real, dimension(n), intent(in) x,
            real, dimension(n), intent(in) dx,
            real, dimension(np), intent(in) param,
            integer, intent(in) n,
            integer, intent(in) np )
```

No premature return.

Author

MH, 2013-10-08

Parameters

in	*n*	Dimension of X
in	*np*	Dimension of param
in	*x*	Vapour mole numbers [mole]
in	*param*	Parameter vector

Returns

Terminate minimization?

5.39.2.10 premterm_at_dx_zero()

```
logical function, public nonlinear_solvers::premterm_at_dx_zero (
            real, dimension(n), intent(in) var,
            real, dimension(n), intent(inout) dvar,
            real, dimension(np), intent(in) param,
            integer, intent(in) n,
            integer, intent(in) np )
```

Terminate search when step size is zero.

Author

MH, 2014

Parameters

in	*n*	Dimension of X
in	*np*	Dimension of param
in	*var*	Variable vector
in,out	*dvar*	Change in variable vector
in	*param*	Parameter vector

5.39.2.11 setxv()

```
subroutine, public nonlinear_solvers::setxv (
             integer, intent(in) n,
             integer, intent(in) nparam,
             real, dimension(n), intent(inout) x,
             real, dimension(n), intent(in) dx,
             real, dimension(n), intent(in) xmin,
             real, dimension(n), intent(in) xmax,
             real, dimension(nparam), intent(inout) param,
             real, intent(in) alpha )
```

Set variables.

Author

MH, 2013-02-27

Parameters

in	*nparam*	Problem dimension
in,out	*x*	Variables
in	*dx*	Change in variables
in	*xmax*	Variable limits
in,out	*param*	Parameter vector
in	*alpha*	dX scaling

5.39.3 Variable Documentation

5.39.3.1 ns_newton

```
integer, parameter, public nonlinear_solvers::ns_newton = 1
```

Solver aliases.
NR solver

5.40 optimizers Module Reference

This module contains generic methods for minimizing systems of non-linear equations.

Data Types

- interface error_function
- type optim_param

  A type that contains solver information.

Functions/Subroutines

- subroutine, public optimize (optim, objective, diff, x, param, limit, premterm, getsize, setxvar, error_fun)

  Interface for optimizer.
- subroutine, public setx (n, nparam, x, dx, param, alpha)

  Set variables.
- logical function, public prematurereturn (x, param, of, dofdx)

  No premature return.
- subroutine, public nelmin (fn, n, start, xmin, ynewlo, reqmin, step, konvge, kcount, icount, numres, ifault)

  NELMIN minimizes a function using the Nelder-Mead algorithm.

Variables

- integer, parameter, public no_mod_newton = 1

  Optimizer aliases.

5.40.1 Detailed Description

This module contains generic methods for minimizing systems of non-linear equations.

Todo Add special treatment for n=2 systems. Analytical modification of eigenvalues.

Author

MHA, February 2012

5.40.2 Function/Subroutine Documentation

5.40.2.1 nelmin()

```
subroutine, public optimizers::nelmin (
             real ( kind = 8 ), external fn,
             integer ( kind = 4 ) n,
             real ( kind = 8 ), dimension(n) start,
             real ( kind = 8 ), dimension(n) xmin,
             real ( kind = 8 ) ynewlo,
             real ( kind = 8 ) reqmin,
             real ( kind = 8 ), dimension(n) step,
             integer ( kind = 4 ) konvge,
             integer ( kind = 4 ) kcount,
             integer ( kind = 4 ) icount,
             integer ( kind = 4 ) numres,
             integer ( kind = 4 ) ifault )
```

NELMIN minimizes a function using the Nelder-Mead algorithm.

Author

KEGT

5.40.2.2 optimize()

```
subroutine, public optimizers::optimize (
             type(optim_param), intent(inout) optim,
             real, external objective,
             external diff,
             real, dimension(:), intent(inout) x,
             real, dimension(:), intent(inout) param,
             external limit,
             logical, external premterm,
             integer, external getsize,
             external setxvar,
             procedure(error_function), optional error_fun )
```

Interface for optimizer.

Author

MHA, February 2012

Parameters

in,out	*optim*	Solver information

Parameters

	objective	The non-linear function F(x)
	diff	Function to evaluate the differentials.
	limit	Function to limit search
	premterm	Test for premature termination
in,out	*x*	On entry, initial condition. On exit, solution
in,out	*param*	Paramaters used by objective, diff, limit and premterm
	getsize	Get size of problem (Usually size(x))
	setxvar	Function to update variables
	setxvar	Function to calculate error

5.40.2.3 prematurereturn()

```
logical function, public optimizers::prematurereturn (
            real, dimension(:), intent(in) x,
            real, dimension(:), intent(in) param,
            real, intent(in) of,
            real, dimension(:), intent(in) dofdx )
```

No premature return.

Author

MH, 2013-02-27

Parameters

in	*x*	Vapour mole numbers [mole]
in	*param*	Parameter vector
in	*of*	Objective function value
in	*dofdx*	Differential of objective function

Returns

Terminate minimization?

5.40.2.4 setx()

```
subroutine, public optimizers::setx (
             integer, intent(in) n,
             integer, intent(in) nparam,
             real, dimension(n), intent(inout) x,
             real, dimension(n), intent(in) dx,
             real, dimension(nparam), intent(inout) param,
             real, intent(in) alpha )
```

Set variables.

Author

MH, 2013-02-27

Parameters

in	*nparam*	Problem dimension
in,out	*x*	Variables

Parameters

in	*dx*	Change in variables
in,out	*param*	Parameter vector
in	*alpha*	dX scaling

5.40.3 Variable Documentation

5.40.3.1 no_mod_newton

```
integer, parameter, public optimizers::no_mod_newton = 1
```

Optimizer aliases.
Modefied newton optimizer

5.41 pc_saft_datadb Module Reference

Automatically generated to file pc_saft_datadb.f90 using utility python code pyUtils Time stamp: 2023-09-06T15↩:26:02.894432.

Data Types

- type pc_saft_data

  PURE COMPONENT PARAMETERS. This data structure stores pure component parameters for the PC-SAFT equation of state.
- type pckijdata

  TEMPERATURE-INDEPENDENT INTERACTION PARAMETERS FOR PC-SAFT DISPERSION TERM.

Variables

- type(pc_saft_data), parameter **pccx1** = pc_saft_data(eosidx = eosPC_SAFT, compName = "BUT1OL", m = 2.9614, sigma = 3.5065e-10, eps_depth_divk = 253.29, eps = 21625.917269816575, beta = 0.↩003874630939, assoc_scheme = assoc_scheme_2C, mu = 0., Q = 0., bib_ref = "de Villiers et al. (2011). Doi: 10.1021/ie200521k", ref = "Default/deVilliers2011")
- type(pc_saft_data), parameter **pccx2** = pc_saft_data(eosidx = eosPC_SAFT, compName = "BUT1OL", m = 2.7515, sigma = 3.6189e-10, eps_depth_divk = 259.59, eps = 21156.981578152732, beta = 0.006692, assoc_scheme = assoc_scheme_2B, mu = 0., Q = 0., bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.↩2010.02.004", ref = "Tang_Gross2010")
- type(pc_saft_data), parameter **pccx3** = pc_saft_data(eosidx = eosPC_SAFT, compName = "HEX1OL", m = 3.1542, sigma = 3.8188e-10, eps_depth_divk = 277.24, eps = 24495.238319341257, beta = 0.↩001308996939, assoc_scheme = assoc_scheme_2C, mu = 0., Q = 0., bib_ref = "de Villiers et al. (2011). Doi: 10.1021/ie200521k", ref = "Default/deVilliers2011")
- type(pc_saft_data), parameter **pccx4** = pc_saft_data(eosidx = eosPC_SAFT, compName = "HEX1OL", m = 3.5146, sigma = 3.6735e-10, eps_depth_divk = 262.32, eps = 21109.589141229262, beta = 0.005747, assoc_scheme = assoc_scheme_2B, mu = 0., Q = 0., bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.↩2010.02.004", ref = "Tang_Gross2010")
- type(pc_saft_data), parameter **pccx5** = pc_saft_data(eosidx = eosPC_SAFT, compName = "PENT1OL", m = 3.1488, sigma = 3.635e-10, eps_depth_divk = 261.96, eps = 21246.2789066717, beta = 0.003926990817, assoc_scheme = assoc_scheme_2C, mu = 0., Q = 0., bib_ref = "de Villiers et al. (2011). Doi: 10.↩1021/ie200521k", ref = "Default/deVilliers2011")
- type(pc_saft_data), parameter **pccx6** = pc_saft_data(eosidx = eosPC_SAFT, compName = "PENT1OL", m = 3.626, sigma = 3.4508e-10, eps_depth_divk = 247.28, eps = 18725.00126234291, beta = 0.010319, assoc↩_scheme = assoc_scheme_2B, mu = 0., Q = 0., bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.↩02.004", ref = "Tang_Gross2010")

- type(pc_saft_data), parameter **pccx7** = pc_saft_data(eosidx = eosPC_SAFT, compName = "PROP1OL", m = 2.9537, sigma = 3.2473e-10, eps_depth_divk = 226.36, eps = 20353.970778491494, beta = 0.↩ 011938052084, assoc_scheme = assoc_scheme_2C, mu = 0., Q = 0., bib_ref = "de Villiers et al. (2011). Doi: 10.1021/ie200521k", ref = "Default/deVilliers2011")
- type(pc_saft_data), parameter **pccx8** = pc_saft_data(eosidx = eosPC_SAFT, compName = "PROP1OL", m = 2.9997, sigma = 3.2522e-10, eps_depth_divk = 233.4, eps = 18930.368489011296, beta = 0.015268, assoc_scheme = assoc_scheme_2B, mu = 0., Q = 0., bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.↩ 2010.02.004", ref = "Tang_Gross2010")
- type(pc_saft_data), parameter **pccx9** = pc_saft_data(eosidx = eosPC_SAFT, compName = "ACETONE", m = 2.7447, sigma = 3.2742e-10, eps_depth_divk = 232.99, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Gross and Vrabec 2006. doi: 10.1002/aic.10683", ref = "Gross2006")
- type(pc_saft_data), parameter **pccx10** = pc_saft_data(eosidx = eosPC_SAFT, compName = "ACETYLENE", m = 1.5477, sigma = 3.3428e-10, eps_depth_divk = 174.48, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Gross 2005, doi: 10.1002/aic.10502", ref = "Gross2005")
- type(pc_saft_data), parameter **pccx11** = pc_saft_data(eosidx = eosPC_SAFT, compName = "ACETYLENE", m = 1.5587, sigma = 3.3325e-10, eps_depth_divk = 174.68, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 4.5415, bib_ref = "Gross 2005, doi: 10.1002/aic.10502", ref = "Gross2005ADJQ")
- type(pc_saft_data), parameter **pccx12** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NH3", m = 2.↩ 25807, sigma = 2.37802e-10, eps_depth_divk = 126.868, eps = 11032.6, beta = 0.20479, assoc_scheme = assoc_scheme_3B, mu = 0., Q = 0., bib_ref = "SINTEF Energy Research", ref = "Default/InHouse")
- type(pc_saft_data), parameter **pccx13** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NH3", m = 1.4302, sigma = 2.7927e-10, eps_depth_divk = 145.0059, eps = 13303.140189, beta = 0.221193, assoc↩ _scheme = assoc_scheme_2B, mu = 0., Q = 0., bib_ref = "10.1016/j.fluid.2020.112689", ref = "Nguyen↩ Huynh2020")
- type(pc_saft_data), parameter **pccx14** = pc_saft_data(eosidx = eosPC_SAFT, compName = "AR", m = 0.↩ 9285, sigma = 3.4784e-10, eps_depth_divk = 122.23, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis↩ _Folas2001")
- type(pc_saft_data), parameter **pccx15** = pc_saft_data(eosidx = eosPC_SAFT, compName = "BENZENE", m = 2.4653, sigma = 3.6478e-10, eps_depth_divk = 287.35, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis_Folas2001")
- type(pc_saft_data), parameter **pccx16** = pc_saft_data(eosidx = eosPC_SAFT, compName = "BENZENE", m = 2.2463, sigma = 3.7852e-10, eps_depth_divk = 296.24, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 5.5907, bib_ref = "Gross 2005, doi: 10.1002/aic.10502", ref = "Gross2005ADJQ")
- type(pc_saft_data), parameter **pccx17** = pc_saft_data(eosidx = eosPC_SAFT, compName = "BUTANAL", m = 2.8825, sigma = 3.4698e-10, eps_depth_divk = 247.09, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Gross and Vrabec (2006). Doi: 10.1002/aic.10683", ref = "Default/Gross2006")
- type(pc_saft_data), parameter **pccx18** = pc_saft_data(eosidx = eosPC_SAFT, compName = "CO2", m = 2.↩ 0729, sigma = 2.7852e-10, eps_depth_divk = 169.21, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis↩ _Folas2001")
- type(pc_saft_data), parameter **pccx19** = pc_saft_data(eosidx = eosPC_SAFT, compName = "CO2", m = 2.↩ 58239, sigma = 2.56106e-10, eps_depth_divk = 151.691, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", ref = "Tang_Gross2010")
- type(pc_saft_data), parameter **pccx20** = pc_saft_data(eosidx = eosPC_SAFT, compName = "CO2", m = 1.72897, sigma = 2.76102e-10, eps_depth_divk = 83.7215, eps = 9256.5, beta = 0.171605, assoc_scheme = assoc_scheme_3B, mu = 0., Q = 0., bib_ref = "Smith (2017) MSc thesis, Doi: 10019.1/101071", ref = "Smith2017")
- type(pc_saft_data), parameter **pccx21** = pc_saft_data(eosidx = eosPC_SAFT, compName = "CO2", m = 1.↩ 5131, sigma = 3.1869e-10, eps_depth_divk = 163.33, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Gross 2005, doi: 10.1002/aic.10502", ref = "Gross2005")
- type(pc_saft_data), parameter **pccx22** = pc_saft_data(eosidx = eosPC_SAFT, compName = "CO2", m = 1.↩ 6298, sigma = 3.0867e-10, eps_depth_divk = 163.34, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 3.9546, bib_ref = "Gross 2005, doi: 10.1002/aic.10502", ref = "Gross2005ADJQ")
- type(pc_saft_data), parameter **pccx23** = pc_saft_data(eosidx = eosPC_SAFT, compName = "CL2", m = 1.↩ 3934, sigma = 3.5339e-10, eps_depth_divk = 270.49, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Gross (2005): doi: 10.1002/aic.10502", ref = "Default/Gross2005")

- type(pc_saft_data), parameter **pccx24** = pc_saft_data(eosidx = eosPC_SAFT, compName = "CL2", m = 1.↩ 4682, sigma = 3.448e-10, eps_depth_divk = 269.67, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 3.0724, bib_ref = "Gross (2005): doi: 10.1002/aic.10502", ref = "Gross2005ADJQ")
- type(pc_saft_data), parameter **pccx25** = pc_saft_data(eosidx = eosPC_SAFT, compName = "CYCLO-HEX", m = 2.5303, sigma = 3.8499e-10, eps_depth_divk = 278.11, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis_Folas2001")
- type(pc_saft_data), parameter **pccx26** = pc_saft_data(eosidx = eosPC_SAFT, compName = "DME", m = 2.2634, sigma = 3.2723e-10, eps_depth_divk = 210.29, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Gross and Vrabeck (2006). Doi: 10.1002/aic.10683", ref = "Default/Gross2006")
- type(pc_saft_data), parameter **pccx27** = pc_saft_data(eosidx = eosPC_SAFT, compName = "C2", m = 1.↩ 6069, sigma = 3.5206e-10, eps_depth_divk = 191.42, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis↩ _Folas2001")
- type(pc_saft_data), parameter **pccx28** = pc_saft_data(eosidx = eosPC_SAFT, compName = "ETOH", m = 2.3609, sigma = 3.1895e-10, eps_depth_divk = 207.56, eps = 22413.213735129506, beta = 0.↩ 017121679962, assoc_scheme = assoc_scheme_2C, mu = 0., Q = 0., bib_ref = "de Villiers et al. (2011). Doi: 10.1021/ie200521k", ref = "Default/deVilliers2011")
- type(pc_saft_data), parameter **pccx29** = pc_saft_data(eosidx = eosPC_SAFT, compName = "ETOH", m = 1.2309, sigma = 4.1057e-10, eps_depth_divk = 316.91, eps = 23372.12070888112, beta = 0.00331438025, assoc_scheme = assoc_scheme_2B, mu = 0., Q = 0., bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.↩ 2010.02.004", ref = "Tang_Gross2010")
- type(pc_saft_data), parameter **pccx30** = pc_saft_data(eosidx = eosPC_SAFT, compName = "ETOH", m = 2.3827, sigma = 3.1771e-10, eps_depth_divk = 198.24, eps = 22061.595111007806, beta = 0.032384, assoc_scheme = assoc_scheme_2B, mu = 0., Q = 0., bib_ref = "Gross & Sadowski (2002). Doi: 10.↩ 1021/ie010954d", ref = "Gross_Sadowski2002")
- type(pc_saft_data), parameter **pccx31** = pc_saft_data(eosidx = eosPC_SAFT, compName = "C2_1", m = 1.5425, sigma = 3.4523e-10, eps_depth_divk = 179.37, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Gross (2005), doi: 10.1002/aic.10502", ref = "Gross2005")
- type(pc_saft_data), parameter **pccx32** = pc_saft_data(eosidx = eosPC_SAFT, compName = "C2_1", m = 1.5477, sigma = 3.4475e-10, eps_depth_divk = 179.19, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 1.9155, bib_ref = "Gross (2005), doi: 10.1002/aic.10502", ref = "Gross2005ADJQ")
- type(pc_saft_data), parameter **pccx33** = pc_saft_data(eosidx = eosPC_SAFT, compName = "H2S", m = 1.↩ 6941, sigma = 3.0214e-10, eps_depth_divk = 226.79, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis↩ _Folas2001")
- type(pc_saft_data), parameter **pccx34** = pc_saft_data(eosidx = eosPC_SAFT, compName = "H2S", m = 1.↩ 355, sigma = 3.309e-10, eps_depth_divk = 234.25, eps = 6491.932412254049, beta = 0.001, assoc_scheme = assoc_scheme_2B, mu = 0., Q = 0., bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", ref = "Default/Tang_Gross2010")
- type(pc_saft_data), parameter **pccx35** = pc_saft_data(eosidx = eosPC_SAFT, compName = "IC4", m = 2.↩ 2616, sigma = 3.7574e-10, eps_depth_divk = 216.53, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis↩ _Folas2001")
- type(pc_saft_data), parameter **pccx36** = pc_saft_data(eosidx = eosPC_SAFT, compName = "IC5", m = 2.↩ 562, sigma = 3.8296e-10, eps_depth_divk = 230.75, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis↩ _Folas2001")
- type(pc_saft_data), parameter **pccx37** = pc_saft_data(eosidx = eosPC_SAFT, compName = "KR", m = 1., sigma = 3.63e-10, eps_depth_divk = 163.1, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Sauer & Gross (2017). DOI: 10.1021/acs.iecr.6b04551", ref = "Default/Sauer_Gross_2017")
- type(pc_saft_data), parameter **pccx38** = pc_saft_data(eosidx = eosPC_SAFT, compName = "C1", m = 1., sigma = 3.7039e-10, eps_depth_divk = 150.03, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis_↩ Folas2001")
- type(pc_saft_data), parameter **pccx39** = pc_saft_data(eosidx = eosPC_SAFT, compName = "MEOH", m = 2.1, sigma = 2.7998e-10, eps_depth_divk = 197.23, eps = 21077.16273701846, beta = 0.430921792317,

assoc_scheme = assoc_scheme_2C, mu = 0., Q = 0., bib_ref = "de Villiers et al. (2011). Doi: 10.↩
1021/ie200521k", ref = "Default/deVilliers2011")

- type(pc_saft_data), parameter **pccx40** = pc_saft_data(eosidx = eosPC_SAFT, compName = "N2", m = 1.↩
2053, sigma = 3.313e-10, eps_depth_divk = 90.96, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0.,
Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis↩
_Folas2001")
- type(pc_saft_data), parameter **pccx41** = pc_saft_data(eosidx = eosPC_SAFT, compName = "N2", m = 1.↩
1504, sigma = 3.3848e-10, eps_depth_divk = 91.4, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0.,
Q = 0., bib_ref = "Gross 2005, doi: 10.1002/aic.10502", ref = "Gross2005")
- type(pc_saft_data), parameter **pccx42** = pc_saft_data(eosidx = eosPC_SAFT, compName = "N2", m = 1.↩
1879, sigma = 3.3353e-10, eps_depth_divk = 90.99, eps = 0., beta = 0., assoc_scheme = no_assoc, mu =
0., Q = 1.1151, bib_ref = "Gross 2005, doi: 10.1002/aic.10502", ref = "Gross2005ADJQ")
- type(pc_saft_data), parameter **pccx43** = pc_saft_data(eosidx = eosPC_SAFT, compName = "O2", m = 1.↩
1217, sigma = 3.2098e-10, eps_depth_divk = 114.96, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0.,
Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis↩
_Folas2001")
- type(pc_saft_data), parameter **pccx44** = pc_saft_data(eosidx = eosPC_SAFT, compName = "C3", m = 2.002,
sigma = 3.6184e-10, eps_depth_divk = 208.11, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q
= 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis_↩
Folas2001")
- type(pc_saft_data), parameter **pccx45** = pc_saft_data(eosidx = eosPC_SAFT, compName = "TOLU",
m = 2.8149, sigma = 3.7169e-10, eps_depth_divk = 285.69, eps = 0., beta = 0., assoc_scheme =
no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref
= "Default/Kontogeorgis_Folas2001")
- type(pc_saft_data), parameter **pccx46** = pc_saft_data(eosidx = eosPC_SAFT, compName = "H2O", m = 1.5,
sigma = 2.6273e-10, eps_depth_divk = 180.3, eps = 15001.119744924437, beta = 0.0942, assoc_scheme
= assoc_scheme_4C, mu = 0., Q = 0., bib_ref = "Grenner et al. (2006). Doi: 10.1021/ie0605332", ref =
"Default/Grenner2006")
- type(pc_saft_data), parameter **pccx47** = pc_saft_data(eosidx = eosPC_SAFT, compName = "H2O", m =
1.0656, sigma = 3.0007e-10, eps_depth_divk = 366.51, eps = 20791.976669215805, beta = 0.034868,
assoc_scheme = assoc_scheme_2B, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.↩
1002/9780470747537", ref = "Kontogeorgis_Folas2001")
- type(pc_saft_data), parameter **pccx48** = pc_saft_data(eosidx = eosPC_SAFT, compName = "H2O", m = 1.↩
18381, sigma = 2.87756e-10, eps_depth_divk = 201.82186, eps = 15074.120726711822, beta = 0.07002,
assoc_scheme = assoc_scheme_4C, mu = 0., Q = 0., bib_ref = "10.1016/j.fluid.2018.06.019", ref = "Nguyen↩
Huynh2020")
- type(pc_saft_data), parameter **pccx49** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC4", m = 2.↩
3316, sigma = 3.7086e-10, eps_depth_divk = 222.88, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0.,
Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis↩
_Folas2001")
- type(pc_saft_data), parameter **pccx50** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC10",
m = 4.6627, sigma = 3.8384e-10, eps_depth_divk = 243.87, eps = 0., beta = 0., assoc_scheme =
no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref
= "Default/Kontogeorgis_Folas2001")
- type(pc_saft_data), parameter **pccx51** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC22",
m = 8.7068, sigma = 3.982e-10, eps_depth_divk = 253.955, eps = 0., beta = 0., assoc_scheme =
no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref
= "Default/Kontogeorgis_Folas2001")
- type(pc_saft_data), parameter **pccx52** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC12", m
= 5.2133, sigma = 3.9115e-10, eps_depth_divk = 248.0042, eps = 0., beta = 0., assoc_scheme =
no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref =
"Default/Kontogeorgis_Folas2001")
- type(pc_saft_data), parameter **pccx53** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC20",
m = 8.0081, sigma = 3.973e-10, eps_depth_divk = 253.1802, eps = 0., beta = 0., assoc_scheme =
no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref =
"Default/Kontogeorgis_Folas2001")

- type(pc_saft_data), parameter **pccx54** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC21", m = 8.3574, sigma = 3.9777e-10, eps_depth_divk = 253.5838, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis_Folas2001")
- type(pc_saft_data), parameter **pccx55** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC17", m = 6.96, sigma = 3.9559e-10, eps_depth_divk = 251.7263, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis_Folas2001")
- type(pc_saft_data), parameter **pccx56** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC7", m = 3.↩ 4831, sigma = 3.8049e-10, eps_depth_divk = 238.4, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis↩ _Folas2001")
- type(pc_saft_data), parameter **pccx57** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC16", m = 6.6107, sigma = 3.949e-10, eps_depth_divk = 251.1392, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis_Folas2001")
- type(pc_saft_data), parameter **pccx58** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC6", m = 3.↩ 0576, sigma = 3.7983e-10, eps_depth_divk = 236.77, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis↩ _Folas2001")
- type(pc_saft_data), parameter **pccx59** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC19", m = 7.6587, sigma = 3.9678e-10, eps_depth_divk = 252.7398, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis_Folas2001")
- type(pc_saft_data), parameter **pccx60** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC9", m = 4.↩ 2079, sigma = 3.8448e-10, eps_depth_divk = 244.51, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis↩ _Folas2001")
- type(pc_saft_data), parameter **pccx61** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC18", m = 7.3094, sigma = 3.9622e-10, eps_depth_divk = 252.2573, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis_Folas2001")
- type(pc_saft_data), parameter **pccx62** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC8", m = 3.↩ 8176, sigma = 3.8373e-10, eps_depth_divk = 242.78, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis↩ _Folas2001")
- type(pc_saft_data), parameter **pccx63** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC25", m = 9.7548, sigma = 3.9931e-10, eps_depth_divk = 254.9091, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis_Folas2001")
- type(pc_saft_data), parameter **pccx64** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC15", m = 6.2614, sigma = 3.9412e-10, eps_depth_divk = 250.4867, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis_Folas2001")
- type(pc_saft_data), parameter **pccx65** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC5", m = 2.↩ 6896, sigma = 3.7729e-10, eps_depth_divk = 231.2, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis↩ _Folas2001")
- type(pc_saft_data), parameter **pccx66** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC14", m = 5.912, sigma = 3.9326e-10, eps_depth_divk = 249.757, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis_Folas2001")
- type(pc_saft_data), parameter **pccx67** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC24", m = 9.4055, sigma = 3.9897e-10, eps_depth_divk = 254.6147, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis_Folas2001")

- type(pc_saft_data), parameter **pccx68** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC23", m = 9.0561, sigma = 3.986e-10, eps_depth_divk = 254.2975, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis_Folas2001")
- type(pc_saft_data), parameter **pccx69** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC13", m = 5.5627, sigma = 3.9227e-10, eps_depth_divk = 248.9356, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis_Folas2001")
- type(pc_saft_data), parameter **pccx70** = pc_saft_data(eosidx = eosPC_SAFT, compName = "NC11", m = 4.864, sigma = 3.8986e-10, eps_depth_divk = 246.939, eps = 0., beta = 0., assoc_scheme = no_assoc, mu = 0., Q = 0., bib_ref = "Kontogeorgis & Folas (2010). Doi: 10.1002/9780470747537", ref = "Default/Kontogeorgis_Folas2001")
- integer, parameter **npcmodels** = 70
- type(pc_saft_data), dimension(npcmodels), parameter **pcarray** = (/ PCcx1,PCcx2,PCcx3,PCcx4,PCcx5, PCcx6,PCcx7,PCcx8,PCcx9,PCcx10, PCcx11,PCcx12,PCcx13,PCcx14,PCcx15, PCcx16,PCcx17,PCcx18,PCcx19,PCcx20, PCcx21,PCcx22,PCcx23,PCcx24,PCcx25, PCcx26,PCcx27,PCcx28,PCcx29,PCcx30, PCcx31,PCcx32,PCcx33,PCcx34,PCcx PCcx36,PCcx37,PCcx38,PCcx39,PCcx40, PCcx41,PCcx42,PCcx43,PCcx44,PCcx45, PCcx46,PCcx47,PCcx48,PCcx49,PCcx PCcx51,PCcx52,PCcx53,PCcx54,PCcx55, PCcx56,PCcx57,PCcx58,PCcx59,PCcx60, PCcx61,PCcx62,PCcx63,PCcx64,PCcx PCcx66,PCcx67,PCcx68,PCcx69,PCcx70 /)
- type(pckijdata), parameter **pcsaft_kij_1** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "H2S", uid2 = "C1", kijvalue = 0.↩ 0425, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_2** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "H2S", uid2 = "C2", kijvalue = 0.072, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_3** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "H2S", uid2 = "C3", kijvalue = 0.069, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_4** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "H2S", uid2 = "NC4", kijvalue = 0.↩ 067, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_5** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "H2S", uid2 = "NC5", kijvalue = 0.↩ 073, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_6** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "H2S", uid2 = "NC6", kijvalue = 0.↩ 073, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_7** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "H2S", uid2 = "NC7", kijvalue = 0.↩ 078, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_8** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "H2S", uid2 = "NC9", kijvalue = 0.↩ 086, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_9** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "H2S", uid2 = "NC10", kijvalue = 0.077, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_10** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "H2S", uid2 = "IC4", kijvalue = 0.06, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_11** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "H2S", uid2 = "IC5", kijvalue = 0.076, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_12** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "H2S", uid2 = "CYCLOHEX", kijvalue = 0.082, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_13** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "H2S", uid2 = "NC8", kijvalue = 0., eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)

- type(pckijdata), parameter **pcsaft_kij_14** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "CO2", uid2 = "H2S", kijvalue = 0.0223, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_15** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "CO2", uid2 = "BENZENE", kijvalue = 0.025, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_16** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "CO2", uid2 = "TOLU", kijvalue = 0.026, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_17** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Gross_↩ Sadowski2001", bib_ref = "Gross & Sadowski (2001). Doi: 10.1021/ie0003887", uid1 = "CO2", uid2 = "C1", kijvalue = 0.065, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_18** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "CO2", uid2 = "C2", kijvalue = 0.102, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_19** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "CO2", uid2 = "C3", kijvalue = 0.0107, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_20** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "CO2", uid2 = "NC4", kijvalue = 0.109, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_21** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "CO2", uid2 = "NC5", kijvalue = 0.12, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_22** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "CO2", uid2 = "NC6", kijvalue = 0.123, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_23** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "CO2", uid2 = "NC7", kijvalue = 0.115, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_24** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "CO2", uid2 = "NC8", kijvalue = 0.132, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_25** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "CO2", uid2 = "NC9", kijvalue = 0.122, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_26** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "CO2", uid2 = "NC10", kijvalue = 0.133, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_27** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "CO2", uid2 = "IC4", kijvalue = 0.112, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_28** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "CO2", uid2 = "IC5", kijvalue = 0.116, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_29** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "CO2", uid2 = "CYCLOHEX", kijvalue = 0.125, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_30** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "CO2", uid2 = "BENZENE", kijvalue = 0.087, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_31** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Tang_↩ Gross2010", bib_ref = "Tang & Gross (2010). Doi: 10.1016/j.fluid.2010.02.004", uid1 = "CO2", uid2 = "TOLU", kijvalue = 0.108, eps_comb_rule = defaultComb, beta_comb_rule = defaultComb)
- type(pckijdata), parameter **pcsaft_kij_32** = PCkijdata(eosidx = eosPC_SAFT, ref = "Default/Nguyen↩ Huynh2020", bib_ref = "Doi: 10.1016/j.fluid.2020.112689", uid1 = "NH3", uid2 = "H2O", kijvalue = -0.32, eps_comb_rule = ariComb, beta_comb_rule = geoComb)

- integer, parameter **pcmaxkij** = 32
- type(pckijdata), dimension(pcmaxkij), parameter **pckijdb** = (/ PCSAFT_KIJ_1,PCSAFT_KIJ_2,PCSAFT_KIJ_3,PCSAFT_KIJ_4,PCSAFT_KIJ_5, PCSAFT_KIJ_6,PCSAFT_KIJ_7,PCSAFT_KIJ_8,PCSAFT_KIJ_9,PCSAFT_KIJ_10, PCSAFT_KIJ_11,PCSAFT_KIJ_12,PCSAFT_KIJ_13,PCSAFT_KIJ_14,PCSAFT_KIJ_15, PCSAFT_KIJ_16,PCSAFT_KIJ_17,PCSAFT_KIJ_18,PCSAFT_KIJ_19,PCSAFT_KIJ_20, PCSAFT_KIJ_21,PCSAFT_KIJ_22,PCSAFT_KIJ_23,PCSAFT_KIJ_24,PCSAFT_KIJ_25, PCSAFT_KIJ_26,PCSAFT_KIJ_27,PCSAFT_KIJ_28,PCSAFT_KIJ_29,PCSAFT_KIJ_30, PCSAFT_KIJ_31,PCSAFT_KIJ_32 /)

5.41.1 Detailed Description

Automatically generated to file pc_saft_datadb.f90 using utility python code pyUtils Time stamp: 2023-09-06T15:26:02.894432.

5.42 pc_saft_nonassoc Module Reference

The module implementing the alpha^{hardchain} and alpha^{dispersion} contributions in PC-SAFT. Parameters are stored in the module pc_saft_parameters, while the association contribution is alpha^{assoc} is implemented in the module saft.

Data Types

- type pcsaft_eos
- type spcsaft_eos

Functions/Subroutines

- subroutine f_spc_saft_tvn (eos, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)

  Gives the contribution to the reduced, residual Helmholtz function F [mol] coming from PC-SAFT's hard-chain and dispersion contributions. All variables are in base SI units. F is defined by F(T,V,n) = sumn∗alpha_PC(rho,T,n) = sumn∗alpha_PC(sumn/V,T,n)
- subroutine alpha_pc (eos, rho, t, n, alp, alp_rho, alp_t, alp_n, alp_rhorho, alp_rhot, alp_rhon, alp_tt, alp_tn, alp_nn)

  alpha_PC = alp^{hard_chain} + alpha^{dispersion}
- subroutine alpha_disp (eos, rho, t, n, alp, alp_rho, alp_t, alp_n, alp_rhorho, alp_rhot, alp_rhon, alp_tt, alp_tn, alp_nn)

  The reduced, molar Helmholtz energy contribution from dispersion.
- subroutine alpha_disp_pc_tvn (eos, t, v, n, alp, alp_v, alp_t, alp_n, alp_vv, alp_tv, alp_vn, alp_tt, alp_tn, alp_nn)

  alpha^{dispersion} TVn alpha = A/(nRT)
- subroutine f_disp_pc_tvn (eos, t, v, n, f, f_v, f_t, f_n, f_vv, f_tv, f_vn, f_tt, f_tn, f_nn)

  F = A/(RT)
- subroutine alpha_spc_saft_hc (eos, rho, t, n, alp, alp_rho, alp_t, alp_n, alp_rhorho, alp_rhot, alp_rhon, alp_tt, alp_tn, alp_nn)
- subroutine alpha_hs_spc_tvn (eos, t, v, n, alp, alp_v, alp_t, alp_n, alp_vv, alp_tv, alp_vn, alp_tt, alp_tn, alp_nn)

  alpha^{hs} TVn alpha = A/(nRT)
- subroutine alpha_spc_saft_hs (eos, rho, t, n, alp, alp_rho, alp_t, alp_n, alp_rhorho, alp_rhot, alp_rhon, alp_tt, alp_tn, alp_nn)
- subroutine g_spc_saft_tvn (eos, t, v, n, g, g_t, g_v, g_n, g_tt, g_tv, g_tn, g_vv, g_vn, g_nn)
- subroutine g_ij_spc_saft (eos, rho, t, n, g, g_rho, g_t, g_n, g_rhorho, g_rhot, g_rhon, g_tt, g_tn, g_nn)
- subroutine m2e2s3_mean (eos, t, n, m2e2s3, m2e2s3_t, m2e2s3_n, m2e2s3_tt, m2e2s3_tn, m2e2s3_nn)

  Equation A.13 of the PC-SAFT article (doi: 10.1021/ie0003887) See also PC-SAFT implementation memo in doc folder.
- subroutine m2e1s3_mean (eos, t, n, m2e1s3, m2e1s3_t, m2e1s3_n, m2e1s3_tt, m2e1s3_tn, m2e1s3_nn)

  Equation A.12 of the PC-SAFT article (doi: 10.1021/ie0003887) See also PC-SAFT implementation memo in doc folder.

- subroutine i_1 (eos, rho, t, n, i1, i1_rho, i1_t, i1_n, i1_rhorho, i1_rhot, i1_rhon, i1_tt, i1_tn, i1_nn)

  A power series approximation of a perturbation theory integral. Equation A.16 of the PC-SAFT article (doi: 10.↩ 1021/ie0003887) See also PC-SAFT implementation memo in doc folder.
- subroutine i_2 (eos, rho, t, n, i2, i2_rho, i2_t, i2_n, i2_rhorho, i2_rhot, i2_rhon, i2_tt, i2_tn, i2_nn)

  A power series approximation of a perturbation theory integral. Equation A.17 of the PC-SAFT article (doi: 10.↩ 1021/ie0003887) See also PC-SAFT implementation memo in doc folder.
- subroutine a_i (eos, n, a, a_n, a_nn)

  The quantities a_i(m_bar) and its derivatives. This is the only routine which accesses a_mat directly. Equation A.18 of the PC-SAFT article (doi: 10.1021/ie0003887) See also PC-SAFT implementation memo in doc folder.
- subroutine b_i (eos, n, b, b_n, b_nn)

  The quantities b_i(m_bar) and its derivatives. The only routine which accesses b_mat directly. Equation A.19 of the PC-SAFT article (doi: 10.1021/ie0003887) See also PC-SAFT implementation memo in doc folder.
- subroutine m_bar (eos, n, mbar, mbar_n, mbar_nn)

  Mean mixture segment number. Equation A.5 of the PC-SAFT article (doi: 10.1021/ie0003887) See also PC-SAFT implementation memo in doc folder.
- subroutine c_1 (eos, rho, t, n, c1, c1_rho, c1_t, c1_n, c1_rhorho, c1_rhot, c1_rhon, c1_tt, c1_tn, c1_nn)

  The compressibility term, defined as (1 + Z^{hc} + rho∗dZ^{hc}/drho)^{-1}.
- subroutine zeta (eos, rho, t, n, z, z_rho, z_t, z_n, z_rhorho, z_rhot, z_rhon, z_tt, z_tn, z_nn)

  Calculates the functions zeta(0),...,zeta(3). zeta(3) equals eta, the packing fraction. Equation A.8 of the PC-SAFT article (doi: 10.1021/ie0003887) See also PC-SAFT implementation memo in doc folder.
- subroutine eta (eos, rho, t, n, e, e_rho, e_t, e_n, e_rhorho, e_rhot, e_rhon, e_tt, e_tn, e_nn)
- subroutine calc_d (eos, t, d, d_t, d_tt)
- subroutine calc_d_hd (eos, t, d)
- subroutine calc_dhs (eos, t)
- subroutine g_pc_saft_tvn (eos, t, v, n, i, j, g, g_t, g_v, g_n, g_tt, g_tv, g_tn, g_vv, g_vn, g_nn)
- subroutine f_hs_pc_saft_tvn (eos, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)

  Gives the contribution to the reduced, residual Helmholtz function F [mol] coming from PC-SAFT's hard-sphere. All variables are in base SI units. F is defined by F(T,V,n) = sumn∗alpha_PC(rho,T,n) = sumn∗alpha_PC(sumn/V,T,n) zeta must be updated!!!!
- subroutine f_chain_pc_saft_tvn (eos, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)

  Gives the contribution to the reduced, residual Helmholtz function F [mol] coming from PC-SAFT's chain term. All variables are in base SI units. F is defined by F(T,V,n) = sumn∗alpha_PC(rho,T,n) = sumn∗alpha_PC(sumn/V,T,n) dhs and zeta must be updated!!!!
- subroutine f_hc_pc_saft_tvn (eos, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)

  Gives the contribution to the reduced, residual Helmholtz function F [mol] coming from PC-SAFT's HC term. All variables are in base SI units. F is defined by F(T,V,n) = sumn∗alpha_PC(rho,T,n) = sumn∗alpha_PC(sumn/V,T,n) dhs and zeta must be updated!!!!
- subroutine alpha_hs_pc_tvn (eos, t, v, n, alp, alp_v, alp_t, alp_n, alp_vv, alp_tv, alp_vn, alp_tt, alp_tn, alp_nn)

  alpha^{hs} TVn alpha = A/(nRT)
- subroutine f_pc_saft_tvn (eos, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)

  Gives the contribution to the reduced, residual Helmholtz function F [mol] coming from PC-SAFT's hard-chain and dispersion contributions. All variables are in base SI units. F is defined by F(T,V,n) = sumn∗alpha_PC(rho,T,n) = sumn∗alpha_PC(sumn/V,T,n)
- subroutine lng_ii_pc_saft_tvn (t, v, n, i, lng, lng_t, lng_v, lng_n, lng_tt, lng_tv, lng_tn, lng_vv, lng_vn, lng_nn)

  Get ln(g_ii) with differentials Boublik (doi: 10.1063/1.1673824) RDF at contact for molecule i in a mixture.
- subroutine spcsaft_allocate_and_init (eos, nc, eos_label)
- subroutine **assign_spcsaft** (this, other)
- subroutine **spcsaft_dealloc** (eos)
- subroutine pcsaft_allocate_and_init (eos, nc, eos_label)
- subroutine **assign_pcsaft** (this, other)
- subroutine **pcsaft_dealloc** (eos)
- class(pcsaft_eos) function, pointer **get_pcsaft_eos_pointer** (base_eos)

Variables

- real, dimension(0:2, 0:6), parameter **a_mat** = reshape((/ 0.9105631445, -0.3084016918, -0.0906148351, 0.6361281449, 0.1860531159, 0.4527842806, 2.6861347891, -2.5030047259, 0.5962700728, -26.↩ 547362491, 21.419793629, -1.7241829131, 97.759208784, -65.255885330, -4.1302112531, -159.↩ 59154087, 83.318680481, 13.776631870, 91.297774084, -33.746922930, -8.6728470368 /), (/3,7/))
- real, dimension(0:2, 0:6), parameter **b_mat** = reshape((/ 0.7240946941, -0.5755498075, 0.0976883116, 2.2382791861, 0.6995095521, -0.2557574982, -4.0025849485, 3.8925673390, -9.1558561530, -21.↩ 003576815, -17.215471648, 20.642075974, 26.855641363, 192.67226447, -38.804430052, 206.55133841, -161.82646165, 93.626774077, -355.60235612, -165.20769346, -29.666905585 /), (/3,7/))
- logical **enable_hs** = .true.
- logical **enable_disp** = .true.

5.42.1 Detailed Description

The module implementing the alpha^{hardchain} and alpha^{dispersion} contributions in PC-SAFT. Parameters are stored in the module pc_saft_parameters, while the association contribution is alpha^{assoc} is implemented in the module saft.
We have implemented the variant sPC-SAFT (simplified PC-SAFT), which has the same (3 or 5) pure-component parameters as PC-SAFT but is built upon a simpler and computationally faster mixing rule.

5.42.2 Function/Subroutine Documentation

5.42.2.1 a_i()

```
subroutine pc_saft_nonassoc::a_i (
             class(spcsaft_eos), intent(in) eos,
             real, dimension(nce), intent(in) n,
             real, dimension(0:6), intent(out) a,
             real, dimension(0:6,nce), intent(out), optional a_n,
             real, dimension(0:6,nce,nce), intent(out), optional a_nn )
```

The quantities a_i(m_bar) and its derivatives. This is the only routine which accesses a_mat directly. Equation A.18 of the PC-SAFT article (doi: 10.1021/ie0003887) See also PC-SAFT implementation memo in doc folder.

Parameters

in	*n*	[mol]
out	*a*	[-]

5.42.2.2 alpha_disp()

```
subroutine pc_saft_nonassoc::alpha_disp (
             class(spcsaft_eos), intent(in) eos,
             real, intent(in) rho,
             real, intent(in) t,
             real, dimension(nce), intent(in) n,
             real, intent(out), optional alp,
             real, intent(out), optional alp_rho,
             real, intent(out), optional alp_t,
             real, dimension(nce), intent(out), optional alp_n,
             real, intent(out), optional alp_rhorho,
             real, intent(out), optional alp_rhot,
             real, dimension(nce), intent(out), optional alp_rhon,
             real, intent(out), optional alp_tt,
             real, dimension(nce), intent(out), optional alp_tn,
             real, dimension(nce,nce), intent(out), optional alp_nn )
```

The reduced, molar Helmholtz energy contribution from dispersion.

Parameters

in	*n*	[mol/m^3], [K], [mol]

5.42.2.3 alpha_disp_pc_tvn()

```
subroutine pc_saft_nonassoc::alpha_disp_pc_tvn (
            class(spcsaft_eos), intent(in) eos,
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(nce), intent(in) n,
            real, intent(out), optional alp,
            real, intent(out), optional alp_v,
            real, intent(out), optional alp_t,
            real, dimension(nce), intent(out), optional alp_n,
            real, intent(out), optional alp_vv,
            real, intent(out), optional alp_tv,
            real, dimension(nce), intent(out), optional alp_vn,
            real, intent(out), optional alp_tt,
            real, dimension(nce), intent(out), optional alp_tn,
            real, dimension(nce,nce), intent(out), optional alp_nn )
```

alpha^{dispersion} TVn alpha = A/(nRT)

Parameters

in	*n*	[m^3], [K], [mol]
out	*alp*	[-]

5.42.2.4 alpha_hs_pc_tvn()

```
subroutine pc_saft_nonassoc::alpha_hs_pc_tvn (
            class(pcsaft_eos), intent(inout) eos,
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(nce), intent(in) n,
            real, intent(out), optional alp,
            real, intent(out), optional alp_v,
            real, intent(out), optional alp_t,
            real, dimension(nce), intent(out), optional alp_n,
            real, intent(out), optional alp_vv,
            real, intent(out), optional alp_tv,
            real, dimension(nce), intent(out), optional alp_vn,
            real, intent(out), optional alp_tt,
            real, dimension(nce), intent(out), optional alp_tn,
            real, dimension(nce,nce), intent(out), optional alp_nn )
```

alpha^{hs} TVn alpha = A/(nRT)

Parameters

in	*n*	[m^3], [K], [mol]
out	*alp*	[-]

5.42.2.5 alpha_hs_spc_tvn()

```
subroutine pc_saft_nonassoc::alpha_hs_spc_tvn (
            class(spcsaft_eos), intent(in) eos,
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(nce), intent(in) n,
            real, intent(out), optional alp,
            real, intent(out), optional alp_v,
            real, intent(out), optional alp_t,
            real, dimension(nce), intent(out), optional alp_n,
            real, intent(out), optional alp_vv,
            real, intent(out), optional alp_tv,
            real, dimension(nce), intent(out), optional alp_vn,
            real, intent(out), optional alp_tt,
            real, dimension(nce), intent(out), optional alp_tn,
            real, dimension(nce,nce), intent(out), optional alp_nn )
```

alpha^{hs} TVn alpha = A/(nRT)

Parameters

in	*n*	[m^3], [K], [mol]
out	*alp*	[-]

5.42.2.6 alpha_pc()

```
subroutine pc_saft_nonassoc::alpha_pc (
            class(spcsaft_eos), intent(in) eos,
            real, intent(in) rho,
            real, intent(in) t,
            real, dimension(nce), intent(in) n,
            real, intent(out), optional alp,
            real, intent(out), optional alp_rho,
            real, intent(out), optional alp_t,
            real, dimension(nce), intent(out), optional alp_n,
            real, intent(out), optional alp_rhorho,
            real, intent(out), optional alp_rhot,
            real, dimension(nce), intent(out), optional alp_rhon,
            real, intent(out), optional alp_tt,
            real, dimension(nce), intent(out), optional alp_tn,
            real, dimension(nce,nce), intent(out), optional alp_nn )
```

alpha_PC = alp^{hard_chain} + alpha^{dispersion}

Parameters

in	*n*	[mol/m^3], [K], [mol]
out	*alp*	[-]

5.42.2.7 alpha_spc_saft_hc()

```
subroutine pc_saft_nonassoc::alpha_spc_saft_hc (
            class(spcsaft_eos), intent(in) eos,
            real, intent(in) rho,
            real, intent(in) t,
            real, dimension(nce), intent(in) n,
            real, intent(out), optional alp,
```

```
            real, intent(out), optional alp_rho,
            real, intent(out), optional alp_t,
            real, dimension(nce), intent(out), optional alp_n,
            real, intent(out), optional alp_rhorho,
            real, intent(out), optional alp_rhot,
            real, dimension(nce), intent(out), optional alp_rhon,
            real, intent(out), optional alp_tt,
            real, dimension(nce), intent(out), optional alp_tn,
            real, dimension(nce,nce), intent(out), optional alp_nn )
```

Parameters

in	*n*	[mol/m^3], [K], [mol]
out	*alp*	[-]

5.42.2.8 alpha_spc_saft_hs()

```
subroutine pc_saft_nonassoc::alpha_spc_saft_hs (
            class(spcsaft_eos), intent(in) eos,
            real, intent(in) rho,
            real, intent(in) t,
            real, dimension(nce), intent(in) n,
            real, intent(out) alp,
            real, intent(out), optional alp_rho,
            real, intent(out), optional alp_t,
            real, dimension(nce), intent(out), optional alp_n,
            real, intent(out), optional alp_rhorho,
            real, intent(out), optional alp_rhot,
            real, dimension(nce), intent(out), optional alp_rhon,
            real, intent(out), optional alp_tt,
            real, dimension(nce), intent(out), optional alp_tn,
            real, dimension(nce,nce), intent(out), optional alp_nn )
```

Parameters

in	*n*	[mol/m^3], [K], [mol]
out	*alp*	[-]

5.42.2.9 b_i()

```
subroutine pc_saft_nonassoc::b_i (
            class(spcsaft_eos), intent(in) eos,
            real, dimension(nce), intent(in) n,
            real, dimension(0:6), intent(out) b,
            real, dimension(0:6,nce), intent(out), optional b_n,
            real, dimension(0:6,nce,nce), intent(out), optional b_nn )
```

The quantities b_i(m_bar) and its derivatives. The only routine which accesses b_mat directly. Equation A.19 of the PC-SAFT article (doi: 10.1021/ie0003887) See also PC-SAFT implementation memo in doc folder.

Parameters

in	*n*	[mol]
out	*b*	[-]

5.42.2.10 c_1()

```
subroutine pc_saft_nonassoc::c_1 (
            class(spcsaft_eos), intent(in) eos,
            real, intent(in) rho,
            real, intent(in) t,
            real, dimension(nce), intent(in) n,
            real, intent(out) c1,
            real, intent(out), optional c1_rho,
            real, intent(out), optional c1_t,
            real, dimension(nce), intent(out), optional c1_n,
            real, intent(out), optional c1_rhorho,
            real, intent(out), optional c1_rhot,
            real, dimension(nce), intent(out), optional c1_rhon,
            real, intent(out), optional c1_tt,
            real, dimension(nce), intent(out), optional c1_tn,
            real, dimension(nce,nce), intent(out), optional c1_nn )
```

The compressibility term, defined as (1 + Z^{hc} + rho∗dZ^{hc}/drho)^{-1}.

Parameters

in	*n*	[mol/m^3], [K], [mol]
out	*c1*	[-]

5.42.2.11 calc_d()

```
subroutine pc_saft_nonassoc::calc_d (
            class(spcsaft_eos), intent(in) eos,
            real, intent(in) t,
            real, dimension(nce), intent(out) d,
            real, dimension(nce), intent(out), optional d_t,
            real, dimension(nce), intent(out), optional d_tt )
```

Parameters

in	*t*	[mol/m^3], [K], [mol]
out	*d*	[m]

5.42.2.12 calc_d_hd()

```
subroutine pc_saft_nonassoc::calc_d_hd (
            class(spcsaft_eos), intent(in) eos,
            type(hyperdual), intent(in) t,
            type(hyperdual), dimension(nce), intent(out) d )
```

Parameters

in	*t*	[mol/m^3], [K], [mol]
out	*d*	[m]

5.42.2.13 calc_dhs()

```
subroutine pc_saft_nonassoc::calc_dhs (
            class(pcsaft_eos), intent(inout) eos,
            real, intent(in) t )
```

Parameters

in	*t*	[K]

5.42.2.14 eta()

```
subroutine pc_saft_nonassoc::eta (
             class(spcsaft_eos), intent(in) eos,
             real, intent(in) rho,
             real, intent(in) t,
             real, dimension(nce), intent(in) n,
             real, intent(out) e,
             real, intent(out), optional e_rho,
             real, intent(out), optional e_t,
             real, dimension(nce), intent(out), optional e_n,
             real, intent(out), optional e_rhorho,
             real, intent(out), optional e_rhot,
             real, dimension(nce), intent(out), optional e_rhon,
             real, intent(out), optional e_tt,
             real, dimension(nce), intent(out), optional e_tn,
             real, dimension(nce,nce), intent(out), optional e_nn )
```

Parameters

in	*n*	[mol/m^3], [K], [mol]
out	*e*	[-]

5.42.2.15 f_chain_pc_saft_tvn()

```
subroutine pc_saft_nonassoc::f_chain_pc_saft_tvn (
             class(pcsaft_eos), intent(inout) eos,
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(nce), intent(in) n,
             real, intent(out), optional f,
             real, intent(out), optional f_t,
             real, intent(out), optional f_v,
             real, dimension(nce), intent(out), optional f_n,
             real, intent(out), optional f_tt,
             real, intent(out), optional f_tv,
             real, dimension(nce), intent(out), optional f_tn,
             real, intent(out), optional f_vv,
             real, dimension(nce), intent(out), optional f_vn,
             real, dimension(nce,nce), intent(out), optional f_nn )
```

Gives the contribution to the reduced, residual Helmholtz function F [mol] coming from PC-SAFT's chain term. All variables are in base SI units. F is defined by F(T,V,n) = sumn∗alpha_PC(rho,T,n) = sumn∗alpha_PC(sumn/V,T,n) dhs and zeta must be updated!!!!

Parameters

out	*f*	[mol]

5.42.2.16 f_disp_pc_tvn()

```
subroutine pc_saft_nonassoc::f_disp_pc_tvn (
```

```
            class(spcsaft_eos), intent(in) eos,
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(nce), intent(in) n,
            real, intent(out), optional f,
            real, intent(out), optional f_v,
            real, intent(out), optional f_t,
            real, dimension(nce), intent(out), optional f_n,
            real, intent(out), optional f_vv,
            real, intent(out), optional f_tv,
            real, dimension(nce), intent(out), optional f_vn,
            real, intent(out), optional f_tt,
            real, dimension(nce), intent(out), optional f_tn,
            real, dimension(nce,nce), intent(out), optional f_nn )
```

F = A/(RT)

Parameters

in	*n*	[m^3], [K], [mol]
out	*f*	[-]

5.42.2.17 f_hc_pc_saft_tvn()

```
subroutine pc_saft_nonassoc::f_hc_pc_saft_tvn (
            class(pcsaft_eos), intent(inout) eos,
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(nce), intent(in) n,
            real, intent(out), optional f,
            real, intent(out), optional f_t,
            real, intent(out), optional f_v,
            real, dimension(nce), intent(out), optional f_n,
            real, intent(out), optional f_tt,
            real, intent(out), optional f_tv,
            real, dimension(nce), intent(out), optional f_tn,
            real, intent(out), optional f_vv,
            real, dimension(nce), intent(out), optional f_vn,
            real, dimension(nce,nce), intent(out), optional f_nn )
```

Gives the contribution to the reduced, residual Helmholtz function F [mol] coming from PC-SAFT's HC term. All variables are in base SI units. F is defined by F(T,V,n) = sumn∗alpha_PC(rho,T,n) = sumn∗alpha_PC(sumn/V,T,n) dhs and zeta must be updated!!!!

Parameters

out	*f*	[mol]

5.42.2.18 f_hs_pc_saft_tvn()

```
subroutine pc_saft_nonassoc::f_hs_pc_saft_tvn (
            class(pcsaft_eos), intent(in) eos,
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(nce), intent(in) n,
            real, intent(out), optional f,
            real, intent(out), optional f_t,
            real, intent(out), optional f_v,
```

```
            real, dimension(nce), intent(out), optional f_n,
            real, intent(out), optional f_tt,
            real, intent(out), optional f_tv,
            real, dimension(nce), intent(out), optional f_tn,
            real, intent(out), optional f_vv,
            real, dimension(nce), intent(out), optional f_vn,
            real, dimension(nce,nce), intent(out), optional f_nn )
```

Gives the contribution to the reduced, residual Helmholtz function F [mol] coming from PC-SAFT's hard-sphere. All variables are in base SI units. F is defined by F(T,V,n) = sumn∗alpha_PC(rho,T,n) = sumn∗alpha_PC(sumn/V,T,n) zeta must be updated!!!!

Parameters

out	*f*	[mol]

5.42.2.19 f_pc_saft_tvn()

```
subroutine pc_saft_nonassoc::f_pc_saft_tvn (
            class(pcsaft_eos), intent(inout) eos,
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(nce), intent(in) n,
            real, intent(out), optional f,
            real, intent(out), optional f_t,
            real, intent(out), optional f_v,
            real, dimension(nce), intent(out), optional f_n,
            real, intent(out), optional f_tt,
            real, intent(out), optional f_tv,
            real, dimension(nce), intent(out), optional f_tn,
            real, intent(out), optional f_vv,
            real, dimension(nce), intent(out), optional f_vn,
            real, dimension(nce,nce), intent(out), optional f_nn )
```

Gives the contribution to the reduced, residual Helmholtz function F [mol] coming from PC-SAFT's hard-chain and dispersion contributions. All variables are in base SI units. F is defined by F(T,V,n) = sumn∗alpha_PC(rho,T,n) = sumn∗alpha_PC(sumn/V,T,n)

Parameters

out	*f*	[mol]

5.42.2.20 f_spc_saft_tvn()

```
subroutine pc_saft_nonassoc::f_spc_saft_tvn (
            class(spcsaft_eos), intent(in) eos,
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(nce), intent(in) n,
            real, intent(out), optional f,
            real, intent(out), optional f_t,
            real, intent(out), optional f_v,
            real, dimension(nce), intent(out), optional f_n,
            real, intent(out), optional f_tt,
            real, intent(out), optional f_tv,
            real, dimension(nce), intent(out), optional f_tn,
            real, intent(out), optional f_vv,
            real, dimension(nce), intent(out), optional f_vn,
```

```
            real, dimension(nce,nce), intent(out), optional f_nn )
```

Gives the contribution to the reduced, residual Helmholtz function F [mol] coming from PC-SAFT's hard-chain and dispersion contributions. All variables are in base SI units. F is defined by F(T,V,n) = sumn∗alpha_PC(rho,T,n) = sumn∗alpha_PC(sumn/V,T,n)

Parameters

out	*f*	[mol]

5.42.2.21 g_ij_spc_saft()

```
subroutine pc_saft_nonassoc::g_ij_spc_saft (
            class(spcsaft_eos), intent(in) eos,
            real, intent(in) rho,
            real, intent(in) t,
            real, dimension(nce), intent(in) n,
            real, intent(out) g,
            real, intent(out), optional g_rho,
            real, intent(out), optional g_t,
            real, dimension(nce), intent(out), optional g_n,
            real, intent(out), optional g_rhorho,
            real, intent(out), optional g_rhot,
            real, dimension(nce), intent(out), optional g_rhon,
            real, intent(out), optional g_tt,
            real, dimension(nce), intent(out), optional g_tn,
            real, dimension(nce,nce), intent(out), optional g_nn )
```

Parameters

in	*n*	[mol/m^3], [K], [mol]
out	*g*	[-]

5.42.2.22 g_pc_saft_tvn()

```
subroutine pc_saft_nonassoc::g_pc_saft_tvn (
            class(pcsaft_eos), intent(inout) eos,
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(nce), intent(in) n,
            integer, intent(in) i,
            integer, intent(in) j,
            real, intent(out) g,
            real, intent(out), optional g_t,
            real, intent(out), optional g_v,
            real, dimension(nce), intent(out), optional g_n,
            real, intent(out), optional g_tt,
            real, intent(out), optional g_tv,
            real, dimension(nce), intent(out), optional g_tn,
            real, intent(out), optional g_vv,
            real, dimension(nce), intent(out), optional g_vn,
            real, dimension(nce,nce), intent(out), optional g_nn )
```

Parameters

in	*n*	[m^3], [K], [mol]
in	*j*	component indices [-]

Parameters

out	*g*	[-]

5.42.2.23 g_spc_saft_tvn()

```
subroutine pc_saft_nonassoc::g_spc_saft_tvn (
             class(spcsaft_eos), intent(in) eos,
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(nce), intent(in) n,
             real, intent(out) g,
             real, intent(out), optional g_t,
             real, intent(out), optional g_v,
             real, dimension(nce), intent(out), optional g_n,
             real, intent(out), optional g_tt,
             real, intent(out), optional g_tv,
             real, dimension(nce), intent(out), optional g_tn,
             real, intent(out), optional g_vv,
             real, dimension(nce), intent(out), optional g_vn,
             real, dimension(nce,nce), intent(out), optional g_nn )
```

Parameters

in	*n*	[m^3], [K], [mol]
out	*g*	[-]

5.42.2.24 i_1()

```
subroutine pc_saft_nonassoc::i_1 (
             class(spcsaft_eos), intent(in) eos,
             real, intent(in) rho,
             real, intent(in) t,
             real, dimension(nce), intent(in) n,
             real, intent(out) i1,
             real, intent(out), optional i1_rho,
             real, intent(out), optional i1_t,
             real, dimension(nce), intent(out), optional i1_n,
             real, intent(out), optional i1_rhorho,
             real, intent(out), optional i1_rhot,
             real, dimension(nce), intent(out), optional i1_rhon,
             real, intent(out), optional i1_tt,
             real, dimension(nce), intent(out), optional i1_tn,
             real, dimension(nce,nce), intent(out), optional i1_nn )
```

A power series approximation of a perturbation theory integral. Equation A.16 of the PC-SAFT article (doi: 10.↩ 1021/ie0003887) See also PC-SAFT implementation memo in doc folder.

Parameters

in	*n*	[mol/m^3], [K], [mol]
out	*i1*	[-]

5.42.2.25 i_2()

```
subroutine pc_saft_nonassoc::i_2 (
            class(spcsaft_eos), intent(in) eos,
            real, intent(in) rho,
            real, intent(in) t,
            real, dimension(nce), intent(in) n,
            real, intent(out) i2,
            real, intent(out), optional i2_rho,
            real, intent(out), optional i2_t,
            real, dimension(nce), intent(out), optional i2_n,
            real, intent(out), optional i2_rhorho,
            real, intent(out), optional i2_rhot,
            real, dimension(nce), intent(out), optional i2_rhon,
            real, intent(out), optional i2_tt,
            real, dimension(nce), intent(out), optional i2_tn,
            real, dimension(nce,nce), intent(out), optional i2_nn )
```

A power series approximation of a perturbation theory integral. Equation A.17 of the PC-SAFT article (doi: 10.↩ 1021/ie0003887) See also PC-SAFT implementation memo in doc folder.

Parameters

in	*n*	[mol/m^3], [K], [mol]
out	*i2*	[-]

5.42.2.26 lng_ii_pc_saft_tvn()

```
subroutine pc_saft_nonassoc::lng_ii_pc_saft_tvn (
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(nce), intent(in) n,
            integer i,
            real, intent(out), optional lng,
            real, intent(out), optional lng_t,
            real, intent(out), optional lng_v,
            real, dimension(nce), intent(out), optional lng_n,
            real, intent(out), optional lng_tt,
            real, intent(out), optional lng_tv,
            real, dimension(nce), intent(out), optional lng_tn,
            real, intent(out), optional lng_vv,
            real, dimension(nce), intent(out), optional lng_vn,
            real, dimension(nce,nce), intent(out), optional lng_nn )
```

Get ln(g_ii) with differentials Boublik (doi: 10.1063/1.1673824) RDF at contact for molecule i in a mixture.

Parameters

out	*lng*	[mol]

5.42.2.27 m2e1s3_mean()

```
subroutine pc_saft_nonassoc::m2e1s3_mean (
            class(spcsaft_eos), intent(in) eos,
            real, intent(in) t,
            real, dimension(nce), intent(in) n,
            real, intent(out) m2e1s3,
            real, intent(out), optional m2e1s3_t,
```

```
            real, dimension(nce), intent(out), optional m2e1s3_n,
            real, intent(out), optional m2e1s3_tt,
            real, dimension(nce), intent(out), optional m2e1s3_tn,
            real, dimension(nce,nce), intent(out), optional m2e1s3_nn )
```

Equation A.12 of the PC-SAFT article (doi: 10.1021/ie0003887) See also PC-SAFT implementation memo in doc folder.

Parameters

in	*n*	[K], [mol]
out	*m2e1s3*	[-]

5.42.2.28 m2e2s3_mean()

```
subroutine pc_saft_nonassoc::m2e2s3_mean (
            class(spcsaft_eos), intent(in) eos,
            real, intent(in) t,
            real, dimension(nce), intent(in) n,
            real, intent(out) m2e2s3,
            real, intent(out), optional m2e2s3_t,
            real, dimension(nce), intent(out), optional m2e2s3_n,
            real, intent(out), optional m2e2s3_tt,
            real, dimension(nce), intent(out), optional m2e2s3_tn,
            real, dimension(nce,nce), intent(out), optional m2e2s3_nn )
```

Equation A.13 of the PC-SAFT article (doi: 10.1021/ie0003887) See also PC-SAFT implementation memo in doc folder.

Parameters

in	*n*	[K], [mol]
out	*m2e2s3*	[-]

5.42.2.29 m_bar()

```
subroutine pc_saft_nonassoc::m_bar (
            class(spcsaft_eos), intent(in) eos,
            real, dimension(nce), intent(in) n,
            real, intent(out) mbar,
            real, dimension(nce), intent(out) mbar_n,
            real, dimension(nce,nce), intent(out) mbar_nn )
```

Mean mixture segment number. Equation A.5 of the PC-SAFT article (doi: 10.1021/ie0003887) See also PC-SAFT implementation memo in doc folder.

Parameters

in	*n*	[mol]
out	*mbar*	[-]
out	*mbar_n*	[1/mol]
out	*mbar_nn*	[1/mol^2]

5.42.2.30 pcsaft_allocate_and_init()

```
subroutine pc_saft_nonassoc::pcsaft_allocate_and_init (
            class(pcsaft_eos), intent(inout) eos,
```

```
            integer, intent(in) nc,
            character(len=*), intent(in) eos_label )
```

Parameters

in	*nc*	Number of components
in	*eos_label*	EOS label

5.42.2.31 spcsaft_allocate_and_init()

```
subroutine pc_saft_nonassoc::spcsaft_allocate_and_init (
            class(spcsaft_eos), intent(inout) eos,
            integer, intent(in) nc,
            character(len=*), intent(in) eos_label )
```

Parameters

in	*nc*	Number of components
in	*eos_label*	EOS label

5.42.2.32 zeta()

```
subroutine pc_saft_nonassoc::zeta (
            class(spcsaft_eos), intent(in) eos,
            real, intent(in) rho,
            real, intent(in) t,
            real, dimension(nce), intent(in) n,
            real, dimension(0:3), intent(out) z,
            real, dimension(0:3), intent(out), optional z_rho,
            real, dimension(0:3), intent(out), optional z_t,
            real, dimension(0:3,nce), intent(out), optional z_n,
            real, dimension(0:3), intent(out), optional z_rhorho,
            real, dimension(0:3), intent(out), optional z_rhot,
            real, dimension(0:3,nce), intent(out), optional z_rhon,
            real, dimension(0:3), intent(out), optional z_tt,
            real, dimension(0:3,nce), intent(out), optional z_tn,
            real, dimension(0:3,nce,nce), intent(out), optional z_nn )
```

Calculates the functions zeta(0),...,zeta(3). zeta(3) equals eta, the packing fraction. Equation A.8 of the PC-SAFT article (doi: 10.1021/ie0003887) See also PC-SAFT implementation memo in doc folder.

Parameters

in	*n*	[mol/m^3], [K], [mol]
out	*z*	[m^(i-3)]

5.43 pc_saft_parameters Module Reference

Module for PC-SAFT pure-component parameters and binary interaction parameters. Also contains parameters for the PeTS equation of state.

Functions/Subroutines

- logical function **rgas_is_correct** ()

Checks that we use the correct gas constant when initializing the pc_saft_data parameters.

- integer function **getpcdataidx** (eosidx, compname, param_ref)

  Get the index in the PCarray of the component having uid given by compName. idx=0 if component isn't in database.
- real function **getpckij** (eosidx, uid1, uid2, param_ref)

  Retrieve binary interaction parameter for components uid1 and uid2. If no kij is stored in the database PCkijdb, it returns 0.0.
- subroutine **getpcsaftkij_allcomps** (nc, comp, eosidx, kij, param_ref)
- subroutine **getpcsaftcombrules_allcomps** (nc, comp, eosidx, epsbeta_combrules, param_ref)
- subroutine **getpcsaftcombrules** (eosidx, uid1, uid2, param_ref, found, epsbetacombrules)

  Retrieve association combining rules for components uid1 and uid2. Found is true if and only if the parameters are in the database.
- subroutine **getpcsaftpureparams_allcomps** (nc, comp, eosidx, param_ref, found, m, sigma, eps_depth_↩ divk, eps, beta, scheme, mu, q)
- subroutine **getpcsaftpureparams_singlecomp** (compname, eosidx, param_ref, found, m, sigma, eps_↩ depth_divk, eps, beta, scheme, mu, q)

5.43.1 Detailed Description

Module for PC-SAFT pure-component parameters and binary interaction parameters. Also contains parameters for the PeTS equation of state.
NB: If you want to add new parameters here, beware that different authors are inconsistent wrt. whether beta should be multiplied with pi/6 or not. For this reason you should validate your results before using these parameters. If you get strange results, try multiplying beta with pi/6=0.5236.

5.44 ph_solver Module Reference

Solve the two-phase PH flash.

Functions/Subroutines

- subroutine, public twophasephflash (t, p, z, beta, betal, x, y, hspec, phase, ierr_out)

  Interface for PH flash.
- subroutine, public singlecomponenttwophasephflash (t, p, z, beta, betal, hspec, tmin, tmax, phase, ierr)

  Do single component PH-flash.
- subroutine, public singlephasepxflash (t, p, z, beta, betal, xspec, tmin, tmax, phase, mode, ierr)

  Do PH/PS flash, assuming we only have one phase.
- subroutine, public setphtolerance (tol)

  Set PH-flash tolerance Caution, write to module variable, and is not thread safe.
- real function, public getphtolerance ()

  Get PH-flash tolerance.

Variables

- integer, parameter, public **ph_mode** =1

  Flash-mode. Some functions allow multiple specifications.
- integer, parameter, public **ps_mode** =2

5.44.1 Detailed Description

Solve the two-phase PH flash.

5.44.2 Function/Subroutine Documentation

5.44.2.1 getphtolerance()

```
real function, public ph_solver::getphtolerance
```

Get PH-flash tolerance.

Author

MH, 2015-03

Returns

Tolerance of hp-flash [-]

5.44.2.2 setphtolerance()

```
subroutine, public ph_solver::setphtolerance (
            real, intent(in) tol )
```

Set PH-flash tolerance Caution, write to module variable, and is not thread safe.

Author

MH, 2015-03

Parameters

in	*tol*	Tolerance for hp-flash [-]

5.44.2.3 singlecomponenttwophasephflash()

```
subroutine, public ph_solver::singlecomponenttwophasephflash (
            real, intent(inout) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) z,
            real, intent(inout) beta,
            real, intent(out) betal,
            real, intent(in) hspec,
            real, intent(inout) tmin,
            real, intent(inout) tmax,
            integer, intent(inout) phase,
            integer, intent(out), optional ierr )
```

Do single component PH-flash.

Author

MH, 2014-10-17

Ailo 2016-12-21

Parameters

in,out	*beta*	Vapour phase molar fraction [-]
out	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in,out	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*hspec*	Specified entropy [J/mol]
in,out	*tmax*	Temperature limits [K]
in,out	*phase*	Phase identifier

5.44.2.4 singlephasepxflash()

```
subroutine, public ph_solver::singlephasepxflash (
             real, intent(inout) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             real, intent(inout) beta,
             real, intent(out) betal,
             real, intent(in) xspec,
             real, intent(in) tmin,
             real, intent(in) tmax,
             integer, intent(in) phase,
             integer, intent(in) mode,
             integer, intent(out), optional ierr )
```

Do PH/PS flash, assuming we only have one phase.

Author

Ailo, 2016-12-21

MH, 2018-10

Parameters

in,out	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*z*	Overall molar compozition [-]
in,out	*beta*	Vapour phase molar fraction [-]
out	*betal*	Liquid phase molar fraction [-]
in	*xspec*	Specified enthalpy/entropy [J/mol(/K)]
in	*tmax*	Temperature limits [K]
in	*phase*	Phase identifier

5.44.2.5 twophasephflash()

```
subroutine, public ph_solver::twophasephflash (
             real, intent(inout) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             real, intent(inout) beta,
             real, intent(out) betal,
             real, dimension(nc), intent(inout) x,
             real, dimension(nc), intent(inout) y,
             real, intent(in) hspec,
             integer, intent(inout) phase,
             integer, intent(out), optional ierr_out )
```

Interface for PH flash.

Author

MH, 2012-03-20

Parameters

in,out	*beta*	Vapour phase molar fraction [-]
out	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]

Parameters

in,out	*x*	Liquid molar compozition [-]
in,out	*y*	Vapour molar compozition [-]
in,out	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*hspec*	Specified enthalpy [J/mol]
in,out	*phase*	Phase identifier
out	*ierr_out*	Error flag (ierr==0 means everything vent well. ierr==-2 out of temperature range, ierr==-1 tolerance not met)

5.45 ps_solver Module Reference

Solve PS-flash specification.

Functions/Subroutines

- subroutine, public twophasepsflash (t, p, z, beta, betal, x, y, sspec, phase, ierr_out)

  Do PS-flash using PT-flash in nested loop.
- subroutine, public singlecomponenttwophasepsflash (t, p, z, beta, betal, x, y, sspec, phase, ierr)

  Do single component PS-flash.
- subroutine, public setpstolerance (tol)

  Set PS-flash tolerance Caution, write to module variable, and is not thread safe.
- real function, public getpstolerance ()

  Get PS-flash tolerance.

5.45.1 Detailed Description

Solve PS-flash specification.

Todo Need trace-component functionallity.

5.45.2 Function/Subroutine Documentation

5.45.2.1 getpstolerance()

```
real function, public ps_solver::getpstolerance
```

Get PS-flash tolerance.

Author

MH, 2015-03

Returns

Tolerance of ps-flash [-]

5.45.2.2 setpstolerance()

```
subroutine, public ps_solver::setpstolerance (
            real, intent(in) tol )
```

Set PS-flash tolerance Caution, write to module variable, and is not thread safe.

Author

MH, 2015-03

Parameters

in	*tol*	Tolerance for ps-flash [-]

5.45.2.3 singlecomponenttwophasepsflash()

```
subroutine, public ps_solver::singlecomponenttwophasepsflash (
             real, intent(inout) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             real, intent(inout) beta,
             real, intent(out) betal,
             real, dimension(nc), intent(inout) x,
             real, dimension(nc), intent(inout) y,
             real, intent(in) sspec,
             integer, intent(inout) phase,
             integer, intent(out), optional ierr )
```

Do single component PS-flash.

Author

MH, 2014-10-17

Parameters

in,out	*beta*	Vapour phase molar fraction [-]
out	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in,out	*x*	Liquid molar compozition [-]
in,out	*y*	Vapour molar compozition [-]
in,out	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*sspec*	Specified entropy [J/mol/K]
in,out	*phase*	Phase identifier

5.45.2.4 twophasepsflash()

```
subroutine, public ps_solver::twophasepsflash (
             real, intent(inout) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             real, intent(inout) beta,
             real, intent(out) betal,
             real, dimension(nc), intent(inout) x,
             real, dimension(nc), intent(inout) y,
             real, intent(in) sspec,
             integer, intent(inout) phase,
             integer, intent(out), optional ierr_out )
```

Do PS-flash using PT-flash in nested loop.

Author

MHA, 2012-03-20

Parameters

in,out	*beta*	Vapour phase molar fraction [-]
out	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in,out	*x*	Liquid molar compozition [-]
in,out	*y*	Vapour molar compozition [-]
in,out	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*sspec*	Specified entropy [J/mol/K]
in,out	*phase*	Phase identifier
out	*ierr_out*	Error flag (ierr==0 means everything vent well. ierr==-2 out of temperature range, ierr==-1 tolerance not met)

5.46 saft_association Module Reference

This module handles all data and routines related to association.

Functions/Subroutines

- subroutine **calc_boltzmann_fac** (assoc, t, boltzmann_fac)

  Calculate Boltzmann factor for association energy, with caching.
- subroutine **delta_kl** (eos, nc, delta, delta_t, delta_v, delta_n, delta_tt, delta_tv, delta_tn, delta_vv, delta_vn, delta_nn)

  Assemble Delta^{kl} matrix, and derivatives if wanted. Can be optimized e.g. by not calculating the exponential in every loop iteration.
- subroutine assemble_m_mich_k (assoc, nc, m_mich_k)

  Assemble the m vector from Michelsen paper, holding the number of moles of each association site.
- subroutine k_mich (eos, nc, k_mich_kl, m_opt, delta_opt)

  Computes the K matrix from Michelsen paper. Needed when solving for X.
- subroutine solve_for_x_k (eos, nc, x_k, maxit, tol)

  Compute the value of X_k consistent with (T,V,n) stored in param.
- subroutine **succ_subs** (eos, nc, x, n_iter)
- subroutine **x_derivatives_knowing_x** (eos, nc, x, x_t, x_v, x_n, x_tt, x_tv, x_vv, x_tn, x_vn, x_nn)

  Computes the derivatives of X. Assumes that X is known.
- subroutine q_derivatives_knowing_x (eos, nc, x_k, q, q_t, q_v, q_n, q_x, q_xt, q_xv, q_xn, q_xx, q_tt, q_tv, q_tn, q_vv, q_vn, q_nn, q_xxx, q_xxt, q_xxv, q_xxn, q_xtt, q_xvv, q_xtv, q_xtn, q_xvn, q_xnn, x_calculated)

  This back-end routine computes the necessary Q-derivatives. Note that X_k is an independent variable of this routine, but if one feeds it an X_k calculated from (T,V,n), one should set X_calculated = .true. This routine is valid for general SAFT equations (both CPA and PC-SAFT).
- subroutine **calcfder_assoc** (eos, nc, x_k, f, f_t, f_v, f_n, f_tt, f_tv, f_vv, f_tn, f_vn, f_nn)

  Calculates the association contribution to the reduced, residual Helmholtz energy, along with its derivatives.
- subroutine **assoc_pressure** (eos, nc, x_k, p, dpdv, dpdt, dpdn)

  Gives the association contribution to pressure.
- subroutine fun (resid0, x, param)

  The X-gradient of Q.
- subroutine jac (j, x, param)

  The X-Hessian of Q, modified according to Michelsen.
- subroutine **hess** (jinv, x, param)

  Just a dummy function needed in nonlinear_solve.
- subroutine **limit** (n, x, xmin, xmax, dx, np, lim_param)

Procedure for limiting the step (deny more than 80% reduction of any X_{A_i}-variable)

- subroutine **fun_succ_subst** (eos, x)

  Successive substitution method.
- type(hyperdual) function **q_fmt_hd** (eos, nc, t, n_fmt, xk0, n)
- subroutine **delta_kl_hd** (eos, t, n_fmt, dhs, delta)

  Assemble Delta^{kl} matrix, and derivatives if wanted. Can be optimized e.g. by not calculating the exponential in every loop iteration.

Variables

- integer, parameter **standard** =1
- integer, parameter **elliot** =2
- integer **delta_combrule** = STANDARD

5.46.1 Detailed Description

This module handles all data and routines related to association.

5.46.2 Function/Subroutine Documentation

5.46.2.1 assemble_m_mich_k()

```
subroutine saft_association::assemble_m_mich_k (
            type(association), intent(in) assoc,
            integer, intent(in) nc,
            real, dimension(numassocsites), intent(out) m_mich_k )
```

Assemble the m vector from Michelsen paper, holding the number of moles of each association site.

Parameters

out	*m_mich_k*	Michelsen m vector.

5.46.2.2 fun()

```
subroutine saft_association::fun (
            real, dimension(numassocsites), intent(out) resid0,
            real, dimension(numassocsites), intent(in) x,
            real, dimension(2+nce), intent(in) param )
```

The X-gradient of Q.

Parameters

out	*resid0*	the X-gradient of Q.
in	*x*	the X-vector

5.46.2.3 jac()

```
subroutine saft_association::jac (
            real, dimension(numassocsites,numassocsites), intent(out) j,
            real, dimension(numassocsites), intent(in) x,
            real, dimension(2+nce), intent(in) param )
```

The X-Hessian of Q, modified according to Michelsen.

Parameters

out	*j*	the modified X-Hessian of Q
in	*x*	the X-vector

5.46.2.4 k_mich()

```
subroutine saft_association::k_mich (
            class(base_eos_param), intent(inout) eos,
            integer, intent(in) nc,
            real, dimension(numassocsites,numassocsites), intent(out) k_mich_kl,
            real, dimension(numassocsites), intent(in), optional m_opt,
            real, dimension(numassocsites,numassocsites), intent(in), optional delta_opt )
```

Computes the K matrix from Michelsen paper. Needed when solving for X.

Parameters

in	*m_opt*	Optional input m_mich_k, in case it has already been computed.
in	*delta_opt*	Optional input Delta_kl, in case it has already been computed.

5.46.2.5 q_derivatives_knowing_x()

```
subroutine saft_association::q_derivatives_knowing_x (
            class(base_eos_param), intent(inout) eos,
            integer, intent(in) nc,
            real, dimension(numassocsites), intent(in) x_k,
            real, intent(out), optional q,
            real, intent(out), optional q_t,
            real, intent(out), optional q_v,
            real, dimension(nc), intent(out), optional q_n,
            real, dimension(numassocsites), intent(out), optional q_x,
            real, dimension(numassocsites), intent(out), optional q_xt,
            real, dimension(numassocsites), intent(out), optional q_xv,
            real, dimension(numassocsites,nc), intent(out), optional q_xn,
            real, dimension(numassocsites,numassocsites), intent(out), optional q_xx,
            real, intent(out), optional q_tt,
            real, intent(out), optional q_tv,
            real, dimension(nc), intent(out), optional q_tn,
            real, intent(out), optional q_vv,
            real, dimension(nc), intent(out), optional q_vn,
            real, dimension(nc,nc), intent(out), optional q_nn,
            real, dimension(numassocsites), intent(out), optional q_xxx,
            real, dimension(numassocsites,numassocsites), intent(out), optional q_xxt,
            real, dimension(numassocsites,numassocsites), intent(out), optional q_xxv,
            real, dimension(numassocsites,numassocsites,nc), intent(out), optional q_xxn,
            real, dimension(numassocsites), intent(out), optional q_xtt,
            real, dimension(numassocsites), intent(out), optional q_xvv,
            real, dimension(numassocsites), intent(out), optional q_xtv,
            real, dimension(numassocsites,nc), intent(out), optional q_xtn,
            real, dimension(numassocsites,nc), intent(out), optional q_xvn,
            real, dimension(numassocsites,nc,nc), intent(out), optional q_xnn,
            logical, intent(in), optional x_calculated )
```

This back-end routine computes the necessary Q-derivatives. Note that X_k is an independent variable of this routine, but if one feeds it an X_k calculated from (T,V,n), one should set X_calculated = .true. This routine is valid for general SAFT equations (both CPA and PC-SAFT).

Parameters

in	*x_calculated*	Is X_k = X_k(T,V,n) calculated?

5.46.2.6 solve_for_x_k()

```
subroutine saft_association::solve_for_x_k (
             class(base_eos_param), intent(inout) eos,
             integer, intent(in) nc,
             real, dimension(numassocsites), intent(inout) x_k,
             integer, intent(in), optional maxit,
             real, intent(in), optional tol )
```

Compute the value of X_k consistent with (T,V,n) stored in param.

Parameters

in	*maxit*	Maximum number of iterations.
in	*tol*	Tolerance.

5.47 saft_interface Module Reference

The interface module for SAFT equations of state. Contains all routines a user may wish to call. Also responsible for combining the association and non-association contributions.

Functions/Subroutines

- subroutine saft_type_eos_init (nc, comp, eos, param_ref, silent_init)

  Called from routine init_thermopack in eoslibinit.f90.
- subroutine cpa_set_cubic_params (nc, comp, cbeos, a0_in, b_in, alphaparams_in, alphacorridx_in, kij_in)

  Sets the fitted parameters in the cubic eos.
- subroutine pcsaft_set_nonassoc_params (eos, nc, m_in, sigma_in, eps_depth_divk_in, kij_in)

  Sets the fitted parameters in the non-association part of PC-SAFT.
- subroutine **pets_set_params** (eos, sigma_in, eps_depth_divk_in)

  Set the molecular parameters in the PeTS equation of state.
- subroutine **saft_setassocparams** (assoc, nc, saft_model, assoc_scheme, epsval, betaval, sigmaval, epsbetacombrulespcsaft)

  Set association parameters for PC-SAFT and SAFT-VR Mie.
- subroutine **cpa_setassocparams** (assoc, nc, assoc_scheme, epsval, betaval, epsbetacombrules, epsbetakij)
- subroutine **calcsaftfder_res** (nc, eos, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_vv, f_tn, f_vn, f_nn, xk)

  Calculates the reduced residual Helmholtz energy F (both the association contribution and the underlying equation (e.g. SRK)), together with its derivatives.
- subroutine **calcsaftfder_res_nonassoc** (nc, eos, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_vv, f_tn, f_vn, f_nn)

  Calculates the reduced residual Helmholtz energy F for the non-associationg part, together with its derivatives.
- subroutine saft_total_pressure_assoc_mix (nc, eos, t, v, n, p, dpdv, dpdt, dpdn)

  Front-end procedure giving the combined pressure of the cubic contribution and the association contribution. Only works for mixtures with association.
- subroutine saft_total_pressure (nc, cbeos, t, v, n, p, dpdv, dpdt, dpdn)

  Front-end procedure giving the combined pressure of the cubic contribution and the association contribution.
- subroutine **calc_saft_dispersion** (t, v, n, a, a_t, a_v, a_n, a_tt, a_tv, a_vv, a_tn, a_vn, a_nn)

  Calculates the reduced dispersion contribution to the Helmholtz energy, together with its derivatives.
- subroutine **calc_saft_hard_sphere** (t, v, n, a, a_t, a_v, a_n, a_tt, a_tv, a_vv, a_tn, a_vn, a_nn)

Calculates the reduced molar hard-sphere contribution to the Helmholtz energy, together with its derivatives.

- subroutine **calc_soft_repulsion** (t, v, n, a, a_t, a_v, a_n, a_tt, a_tv, a_vv, a_tn, a_vn, a_nn)

  Calculates the reduced molar soft repulsion contribution to the Helmholtz energy, together with its derivatives.
- subroutine **calc_hard_sphere_diameter** (t, d, d_t)

  Calculates Hard-sphere diameter.
- subroutine **calc_hard_sphere_diameter_ij** (i, j, t, d, d_t)

  Calculates non-additive Hard-sphere diameter.
- subroutine **truncation_corrections** (enable_truncation_correction, enable_shift_correction, reduced_↩ radius_cut)

  Enable/disable truncation corrections.
- subroutine de_broglie_wavelength (i, t, lambda)

  Return de Broglie wavelength for component i.
- subroutine de_boer_parameter (i, lambda)

  Return de Boer parameter for component i.
- subroutine adjust_mass_to_specified_de_boer_parameter (i, lambda)

  Return de Boer parameter for component i.
- subroutine potential (i, j, n, r, t, pot)

  Return interaction potential between component i and j.
- subroutine alpha (t, a_ij)

  Calculate dimensionless van der Waals energy for interaction potentials.
- subroutine epsilon_ij (i, j, eps_div_kb_ij)

  Get well depth divided by kB for interaction i and j.
- subroutine epsilon_eff_ij (i, j, t, eps_div_kb_ij)

  Get effective well depth divided by kB for interaction i and j.
- subroutine sigma_ij (i, j, s_ij)

  Size parameter for interaction i and j.
- subroutine sigma_eff_ij (i, j, t, s_ij)

  Effective size parameter for interaction i and j.
- subroutine test_fmt_compatibility (is_fmt_consistent, na_enabled)

  Test if model setup is comaptible with the Fundamental Measure Theory (FMT)
- subroutine saft_zfac (nc, eos, phase, t, p, n, z, dzdt, dzdp, dzdn)

  Calculate the compressibility and its derivatives.
- subroutine saft_lnphi (nc, eos, phase, t, p, n, lnphi, dlnphidt, dlnphidp, dlnphidn)

  Calculate the logarithmic fugacity and its derivatives.
- subroutine saft_residentropy (nc, eos, phase, t, p, n, s, dsdt, dsdp, dsdn)
- subroutine saft_residenthalpy (nc, eos, phase, t, p, n, h, dhdt, dhdp, dhdn)
- subroutine saft_residgibbs (nc, eos, phase, t, p, n, g, dgdt, dgdp, dgdn)
- subroutine **saft_master_volume_solver** (nc, cbeos, t, p_spec, n, phase, v)
- subroutine **pc_saft_nonassoc_volume_solver** (nc, cbeos, t, p_spec, n, phase, v)

  Routine for when we want to use PC-SAFT on a non-association mixture.
- real function **conversion_numerator** (eos, nc, t, n)

  Calculate conversion numerator for pressure solver.
- subroutine **saft_volume_solver** (nc, eos, t, p_spec, n, phase, v)

  Volume solver for associating mixtures. Modeled after the paper: Michelsen (2006) "Robust and Efficient Solution Procedures for Association Models.".
- subroutine saft_total_pressure_knowing_x_k (nc, eos, x_k, p, dpdv, dpdt, dpdn)

  A back-end procedure giving the combined pressure of the cubic contribution and the association contribution.
- subroutine **nonassoc_pressure** (nc, eos, t, v, n, p, dpdv, dpdt, dpdn)

  The pressure contribution not coming from association.
- subroutine **calcfder_nonassoc_cpa** (nc, cbeos, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_vv, f_tn, f_vn, f_nn)

  Calculates the contibution to the reduced residual Helmholtz energy F coming from the non-association part, along with its derivatives.

- subroutine **compute_dxdv_and_dpdv** (nc, eos, x_k, x_v, p_v)

  Special routine for computing the derivatives needed in the Newton iteration of volume_solver.
- subroutine pc_saft_get_kij (i, j, kij)

  Routine useful when fitting binary interaction parameters.
- subroutine pc_saft_set_kij (i, j, kij)

  Routine useful when fitting binary interaction parameters.
- subroutine pc_saft_set_kij_asym (i, j, kij)

  Routine useful when fitting binary interaction parameters. For the cases when kij/=kji.
- subroutine cpa_get_kij (i, j, aeps_kij_out)

  Routine useful when fitting binary interaction parameters.
- subroutine cpa_set_kij (i, j, aeps_kij_in)

  Routine useful when fitting binary interaction parameters.
- subroutine cpa_set_pure_params (ic, params)

  Input a0, b in their conventional (non-SI) units, beta and eps in SI units, c1 dimensionless.
- subroutine cpa_get_pure_params (ic, params)
- subroutine **pc_saft_set_pure_params** (ic, params)
- subroutine **pc_saft_get_pure_params** (ic, params)
- subroutine **pets_set_pure_params** (ic, params)
- subroutine **pets_get_pure_params** (ic, params)
- subroutine **getactiveassocparams** (assoc, ic, eps, beta)
- subroutine **setactiveassocparams** (assoc, ic, eps, beta)
- subroutine **printbinarymixturereportsaft** ()
- subroutine **setcpaformulation** (simplified)

  Lets the user choose whether to use the simplified or the original formulation of CPA.
- subroutine **estimate_critical_parameters** (i, tc, vc)

  Estimate critical parameters based on reduced values.
- subroutine **calc_assoc_phi** (n_fmt, t, f, f_t, f_n, f_tt, f_tn, f_nn)

  Calculates the reduced association Helmholtz energy density together with its derivatives. FMT interface.
- subroutine **test_calc_assoc_phi** ()

  Test calc_assoc_phi.
- subroutine **set_fmt_densities** (t, v, n, n_fmt)

5.47.1 Detailed Description

The interface module for SAFT equations of state. Contains all routines a user may wish to call. Also responsible for combining the association and non-association contributions.
Available SAFT equations: CPA-SRK, CPA-PR, PC-SAFT and SAFT-VR Mie.
Caveat for future programmers modifying this module: It operates only with SI units. Most notably, the routines use V [m^3] and n [mole numbers] instead of v [L/mol] and z [normalized mole numbers]. However, the a and b parameters in the CPA database use non-SI units that comply with the units in eoscubic type.

5.47.2 Function/Subroutine Documentation

5.47.2.1 adjust_mass_to_specified_de_boer_parameter()

```
subroutine saft_interface::adjust_mass_to_specified_de_boer_parameter (
            integer, intent(in) i,
            real, intent(in) lambda )
```

Return de Boer parameter for component i.

Author

Morten Hammer, July 2022

Parameters

in	*i*	Component number
in	*lambda*	de Boer

5.47.2.2 alpha()

```
subroutine saft_interface::alpha (
            real, intent(in) t,
            real, dimension(nce,nce), intent(out) a_ij )
```

Calculate dimensionless van der Waals energy for interaction potentials.

Author

Morten Hammer, June 2023

Parameters

in	*t*	Temperature
out	*a_ij*	Dimensionless van der Waals energy

5.47.2.3 cpa_get_kij()

```
subroutine saft_interface::cpa_get_kij (
            integer, intent(in) i,
            integer, intent(in) j,
            real, dimension(2), intent(out) aeps_kij_out )
```

Routine useful when fitting binary interaction parameters.

Parameters

in	*j*	Component indices.
out	*aeps_kij_out*	Binary interaction parameters.

5.47.2.4 cpa_get_pure_params()

```
subroutine saft_interface::cpa_get_pure_params (
            integer, intent(in) ic,
            real, dimension(5), intent(out) params )
```

Parameters

out	*params*	a0, b, beta, eps, c1

5.47.2.5 cpa_set_cubic_params()

```
subroutine saft_interface::cpa_set_cubic_params (
            integer, intent(in) nc,
            type(gendata_pointer), dimension(nc), intent(inout) comp,
            class(cb_eos), intent(inout) cbeos,
            real, dimension(nc), intent(in) a0_in,
            real, dimension(nc), intent(in) b_in,
```

```
            real, dimension(3,nc), intent(in) alphaparams_in,
            integer, dimension(nc), intent(in) alphacorridx_in,
            real, dimension(nc,nc), intent(in) kij_in )
```

Sets the fitted parameters in the cubic eos.

Parameters

in	*nc*	Number of components.
in,out	*comp*	Component vector.
in,out	*cbeos*	The underlying cubic equation of state.

5.47.2.6 cpa_set_kij()

```
subroutine saft_interface::cpa_set_kij (
            integer, intent(in) i,
            integer, intent(in) j,
            real, dimension(2), intent(in) aeps_kij_in )
```

Routine useful when fitting binary interaction parameters.

Parameters

in	*j*	Component indices.
in	*aeps_kij_in*	Binary interaction parameters.

5.47.2.7 cpa_set_pure_params()

```
subroutine saft_interface::cpa_set_pure_params (
            integer, intent(in) ic,
            real, dimension(5), intent(in) params )
```

Input a0, b in their conventional (non-SI) units, beta and eps in SI units, c1 dimensionless.

Parameters

in	*params*	a0, b, beta, eps, c1

5.47.2.8 de_boer_parameter()

```
subroutine saft_interface::de_boer_parameter (
            integer, intent(in) i,
            real, intent(out) lambda )
```

Return de Boer parameter for component i.

Author

Morten Hammer, July 2022

Parameters

in	*i*	Component number
out	*lambda*	de Boer

5.47.2.9 de_broglie_wavelength()

```
subroutine saft_interface::de_broglie_wavelength (
             integer, intent(in) i,
             real, intent(in) t,
             real, intent(out) lambda )
```

Return de Broglie wavelength for component i.

Author

Morten Hammer, March 2022

Parameters

in	*i*	Component number
in	*t*	Temperature
out	*lambda*	de Broglie wavelength

5.47.2.10 epsilon_eff_ij()

```
subroutine saft_interface::epsilon_eff_ij (
             integer, intent(in) i,
             integer, intent(in) j,
             real, intent(in) t,
             real, intent(out) eps_div_kb_ij )
```

Get effective well depth divided by kB for interaction i and j.

Author

Morten Hammer, June 2023

Parameters

in	*j*	Component number
in	*t*	Temperature (K)
out	*eps_div_kb_ij*	Effective well depth divided by Boltzmann constant

5.47.2.11 epsilon_ij()

```
subroutine saft_interface::epsilon_ij (
             integer, intent(in) i,
             integer, intent(in) j,
             real, intent(out) eps_div_kb_ij )
```

Get well depth divided by kB for interaction i and j.

Author

Morten Hammer, June 2023

Parameters

in	*j*	Component number
out	*eps_div_kb_ij*	Well depth divided by Boltzmann constant

5.47.2.12 pc_saft_get_kij()

```
subroutine saft_interface::pc_saft_get_kij (
             integer, intent(in) i,
             integer, intent(in) j,
             real, intent(out) kij )
```

Routine useful when fitting binary interaction parameters.

Parameters

in	*j*	Component indices.
out	*kij*	Binary interaction parameter.

5.47.2.13 pc_saft_set_kij()

```
subroutine saft_interface::pc_saft_set_kij (
             integer, intent(in) i,
             integer, intent(in) j,
             real, intent(in) kij )
```

Routine useful when fitting binary interaction parameters.

Parameters

in	*j*	Component indices.
in	*kij*	Binary interaction parameter.

5.47.2.14 pc_saft_set_kij_asym()

```
subroutine saft_interface::pc_saft_set_kij_asym (
             integer, intent(in) i,
             integer, intent(in) j,
             real, intent(in) kij )
```

Routine useful when fitting binary interaction parameters. For the cases when kij/=kji.

Parameters

in	*j*	Component indices.
in	*kij*	Binary interaction parameter.

5.47.2.15 pcsaft_set_nonassoc_params()

```
subroutine saft_interface::pcsaft_set_nonassoc_params (
             class(spcsaft_eos), intent(inout) eos,
             integer, intent(in) nc,
             real, dimension(nc), intent(in) m_in,
             real, dimension(nc), intent(in) sigma_in,
             real, dimension(nc), intent(in) eps_depth_divk_in,
             real, dimension(nc,nc), intent(in) kij_in )
```

Sets the fitted parameters in the non-association part of PC-SAFT.

Parameters

in	*nc*	Number of components.

5.47.2.16 potential()

```
subroutine saft_interface::potential (
             integer, intent(in) i,
             integer, intent(in) j,
             integer, intent(in) n,
             real, dimension(n), intent(in) r,
             real, intent(in) t,
             real, dimension(n), intent(out) pot )
```

Return interaction potential between component i and j.

Author

Morten Hammer, July 2022

Parameters

in	*j*	Component number
in	*t*	Temperature
in	*n*	Array size
in	*r*	Intermolecular separation (m)
out	*pot*	Potential divided by Boltzmann constant

5.47.2.17 saft_lnphi()

```
subroutine saft_interface::saft_lnphi (
             integer, intent(in) nc,
             class (base_eos_param), intent(inout) eos,
             integer, intent(in) phase,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) n,
             real, dimension(nc), intent(out) lnphi,
             real, dimension(nc), intent(out), optional dlnphidt,
             real, dimension(nc), intent(out), optional dlnphidp,
             real, dimension(nc,nc), intent(out), optional dlnphidn )
```

Calculate the logarithmic fugacity and its derivatives.

Parameters

in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*n*	Mole numbers [moles]

5.47.2.18 saft_residenthalpy()

```
subroutine saft_interface::saft_residenthalpy (
             integer, intent(in) nc,
             class(base_eos_param), intent(inout) eos,
             integer, intent(in) phase,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) n,
             real, intent(out) h,
             real, intent(out), optional dhdt,
```

```
            real, intent(out), optional dhdp,
            real, dimension(nc), intent(out), optional dhdn )
```

Parameters

in,out	*eos*	Cubic eos
in	*p*	Pressure [Pa]
in	*t*	Temperature [K]
in	*phase*	Phase identifier [-]
in	*n*	Composition [mol]
out	*h*	Enthalpy [J/mol/K]

5.47.2.19 saft_residentropy()

```
subroutine saft_interface::saft_residentropy (
            integer, intent(in) nc,
            class(base_eos_param), intent(inout) eos,
            integer, intent(in) phase,
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) n,
            real, intent(out) s,
            real, intent(out), optional dsdt,
            real, intent(out), optional dsdp,
            real, dimension(nc), intent(out), optional dsdn )
```

Parameters

in	*nc*	Number of components in mixture.
in,out	*eos*	Cubic eos for
in	*p*	Pressure [Pa]
in	*t*	Temperature [K]
in	*phase*	Phase identifier [-]
in	*n*	Composition [mol]
out	*s*	Entropy [J/mol/K]

5.47.2.20 saft_residgibbs()

```
subroutine saft_interface::saft_residgibbs (
            integer, intent(in) nc,
            class(base_eos_param), intent(inout) eos,
            integer, intent(in) phase,
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) n,
            real, intent(out) g,
            real, intent(out), optional dgdt,
            real, intent(out), optional dgdp,
            real, dimension(nc), intent(out), optional dgdn )
```

Parameters

in,out	*eos*	Cubic eos.
in	*p*	Pressure [Pa]

Parameters

in	*t*	Temperature [K]
in	*phase*	Phase identifier [-]
in	*n*	Composition [mol]
out	*g*	Gibbs free energy [J/mol]

5.47.2.21 saft_total_pressure()

```
subroutine saft_interface::saft_total_pressure (
            integer, intent(in) nc,
            class(base_eos_param), intent(inout) cbeos,
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(nc), intent(in) n,
            real, intent(out) p,
            real, intent(out), optional dpdv,
            real, intent(out), optional dpdt,
            real, dimension(nc), intent(out), optional dpdn )
```

Front-end procedure giving the combined pressure of the cubic contribution and the association contribution.

Parameters

in	*t*	Temperature [K]
in	*v*	Volume [m^3]
in	*n*	Mole numbers [moles]
out	*p*	Pressure [Pa]

5.47.2.22 saft_total_pressure_assoc_mix()

```
subroutine saft_interface::saft_total_pressure_assoc_mix (
            integer, intent(in) nc,
            class(base_eos_param), intent(inout) eos,
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(nc), intent(in) n,
            real, intent(out) p,
            real, intent(out), optional dpdv,
            real, intent(out), optional dpdt,
            real, dimension(nc), intent(out), optional dpdn )
```

Front-end procedure giving the combined pressure of the cubic contribution and the association contribution. Only works for mixtures with association.

Parameters

in	*t*	Temperature [K]
in	*v*	Volume [m^3]
in	*n*	Mole numbers [moles]
out	*p*	Pressure [Pa]

5.47.2.23 saft_total_pressure_knowing_x_k()

```
subroutine saft_interface::saft_total_pressure_knowing_x_k (
```

```
            integer, intent(in) nc,
            class(base_eos_param), intent(inout) eos,
            real, dimension(numassocsites), intent(in) x_k,
            real, intent(out) p,
            real, intent(out), optional dpdv,
            real, intent(out), optional dpdt,
            real, dimension(nc), intent(out), optional dpdn )
```

A back-end procedure giving the combined pressure of the cubic contribution and the association contribution.

Parameters

out	*p*	Pressure [Pa]

5.47.2.24 saft_type_eos_init()

```
subroutine saft_interface::saft_type_eos_init (
            integer, intent(in) nc,
            type(gendata_pointer), dimension(nc), intent(inout) comp,
            class(base_eos_param), intent(inout) eos,
            character(len=*), intent(in) param_ref,
            logical, intent(in), optional silent_init )
```

Called from routine init_thermopack in eoslibinit.f90.

Parameters

in	*nc*	Number of components.
in,out	*comp*	Component vector.
in,out	*eos*	Underlying cubic equation of state.
in	*param_ref*	Parameter sets to use for components
in	*silent_init*	Print no varnings during init

5.47.2.25 saft_zfac()

```
subroutine saft_interface::saft_zfac (
            integer, intent(in) nc,
            class(base_eos_param), intent(inout) eos,
            integer, intent(in) phase,
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) n,
            real, intent(out) z,
            real, intent(out), optional dzdt,
            real, intent(out), optional dzdp,
            real, dimension(nc), intent(out), optional dzdn )
```

Calculate the compressibility and its derivatives.

Parameters

in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*n*	Mole numbers [moles]

5.47.2.26 sigma_eff_ij()

```
subroutine saft_interface::sigma_eff_ij (
             integer, intent(in) i,
             integer, intent(in) j,
             real, intent(in) t,
             real, intent(out) s_ij )
```

Effective size parameter for interaction i and j.

Author

Morten Hammer, June 2023

Parameters

in	*j*	Component number
in	*t*	Temperature (K)
out	s_{ij}	Effective size paramater (m)

5.47.2.27 sigma_ij()

```
subroutine saft_interface::sigma_ij (
             integer, intent(in) i,
             integer, intent(in) j,
             real, intent(out) s_ij )
```

Size parameter for interaction i and j.

Author

Morten Hammer, June 2023

Parameters

in	*j*	Component number
out	s_{ij}	Size paramater (m)

5.47.2.28 test_fmt_compatibility()

```
subroutine saft_interface::test_fmt_compatibility (
             logical, intent(out) is_fmt_consistent,
             logical, intent(out) na_enabled )
```

Test if model setup is comaptible with the Fundamental Measure Theory (FMT)

Author

Morten Hammer, October 2022

5.48 saft_rdf Module Reference

Module responsible for radial distribution functions.

Functions/Subroutines

- subroutine, public master_saft_rdf (eos, nc, i, j, g, g_t, g_v, g_n, g_tt, g_tv, g_tn, g_vv, g_vn, g_nn)

  Radial distribution function (RDF) interface for association.

Variables

- logical, public **usesimplifiedcpa** = .FALSE.

5.48.1 Detailed Description

Module responsible for radial distribution functions.

5.48.2 Function/Subroutine Documentation

5.48.2.1 master_saft_rdf()

```
subroutine, public saft_rdf::master_saft_rdf (
             class(base_eos_param), intent(inout) eos,
             integer, intent(in) nc,
             integer, intent(in) i,
             integer, intent(in) j,
             real, intent(out) g,
             real, intent(out), optional g_t,
             real, intent(out), optional g_v,
             real, dimension(nc), intent(out), optional g_n,
             real, intent(out), optional g_tt,
             real, intent(out), optional g_tv,
             real, dimension(nc), intent(out), optional g_tn,
             real, intent(out), optional g_vv,
             real, dimension(nc), intent(out), optional g_vn,
             real, dimension(nc,nc), intent(out), optional g_nn )
```

Radial distribution function (RDF) interface for association.

Parameters

in,out	*eos*	Depends on component indices i,j only for eosBH_pert
in	*j*	component indices [-]
out	*g*	The rdf gij [-]

5.49 saftvrmie_datadb Module Reference

Automatically generated to file saftvrmie_datadb.f90 using utility python code pyUtils Time stamp: 2023-06-21T13↩:07:27.307865.

Data Types

- type miekijdata

  INTERACTION PARAMETERS FOR THE SAFT-VR-MIE DISPERSION TERM.
- type saftvrmie_data

  PURE COMPONENT PARAMETERS. This data structure stores pure component parameters for the SAFT-VRQ Mie EoS.

Variables

- type(saftvrmie_data), parameter **miecx1** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NH3", m = 1., sigma = 3.3309e-10, eps_depth_divk = 323.7, lambda_a = 6., lambda_r = 36.832, mass = 2.828e-26,

eps = 1105., beta = 5.6073E-28, assoc_scheme = assoc_scheme_4B, fh_order = 0, bib_ref = "Dufal (2015) - 10.1080/00268976.2015.1029027", ref = "DEFAULT/Dufal2015")

- type(saftvrmie_data), parameter **miecx2** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "AR", m = 1., sigma = 3.41e-10, eps_depth_divk = 118.7, lambda_a = 6., lambda_r = 12.26, mass = 6.6335e-26, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Unpublished (as of 08/2018) parameters from G. Jackon's group", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx3** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "CO2", m = 1.5, sigma = 3.1916e-10, eps_depth_divk = 231.88, lambda_a = 5.1646, lambda_r = 27.557, mass = 0.e+00, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Lafitte et al. 2013, doi: 10.1063/1.4819786", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx4** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "D2", m = 1., sigma = 3.1538e-10, eps_depth_divk = 21.2, lambda_a = 6., lambda_r = 8., mass = 6.689e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Aasen 2019, doi: 10.1063/1.↩ 5111364", ref = "DEFAULT/AASEN2019-FH0")
- type(saftvrmie_data), parameter **miecx5** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "D2", m = 1., sigma = 3.0203e-10, eps_depth_divk = 30.273, lambda_a = 6., lambda_r = 10., mass = 6.689e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 1, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH1")
- type(saftvrmie_data), parameter **miecx6** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "D2", m = 1., sigma = 2.9897e-10, eps_depth_divk = 36.913, lambda_a = 6., lambda_r = 12., mass = 6.689e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 2, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH2/AASEN2019-FH2-LJ")
- type(saftvrmie_data), parameter **miecx7** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "D2", m = 1., sigma = 3.1561e-10, eps_depth_divk = 28.222, lambda_a = 6., lambda_r = 12., mass = 6.689e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH0-LJ")
- type(saftvrmie_data), parameter **miecx8** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "D2", m = 1., sigma = 3.0193e-10, eps_depth_divk = 34.389, lambda_a = 6., lambda_r = 12., mass = 6.689e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 1, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH1-LJ")
- type(saftvrmie_data), parameter **miecx9** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "D2", m = 1., sigma = 3.009e-10, eps_depth_divk = 39.239, lambda_a = 7., lambda_r = 11., mass = 6.689e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 1, bib_ref = "Hammer 2022, doi: xxxx", ref = "HAMMER2022-FH1")
- type(saftvrmie_data), parameter **miecx10** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "C2", m = 1.4373, sigma = 3.7257e-10, eps_depth_divk = 206.12, lambda_a = 6., lambda_r = 12.4, mass = 0.e+00, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Lafitte et al. 2013, doi: 10.1063/1.4819786", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx11** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "ETOH", m = 2.25648939, sigma = 3.2903e-10, eps_depth_divk = 238.97, lambda_a = 6., lambda_r = 12.↩ 282, mass = 0.e+00, eps = 2247.3, beta = 4.2794E-29, assoc_scheme = assoc_scheme_3B, fh_order = 0, bib_ref = "Dufal (2015) - 10.1080/00268976.2015.1029027", ref = "DEFAULT/Dufal2015")
- type(saftvrmie_data), parameter **miecx12** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "ETOH", m = 1.7639, sigma = 3.6025e-10, eps_depth_divk = 307.92, lambda_a = 6., lambda_r = 17.968, mass = 0.e+00, eps = 2380., beta = 1.5018E-28, assoc_scheme = assoc_scheme_3B, fh_order = 0, bib_ref = "Polishuk (2018) - 10.1016/j.molliq.2018.05.112", ref = "Polishuk")
- type(saftvrmie_data), parameter **miecx13** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "HE", m = 1., sigma = 3.353e-10, eps_depth_divk = 4.44, lambda_a = 6., lambda_r = 14.84, mass = 6.6464764e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Herdes 2015, doi: 10.1016/j.fluid.2015.07.014", ref = "DEFAULT/AASEN2019-FH0")
- type(saftvrmie_data), parameter **miecx14** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "HE", m = 1., sigma = 2.7443e-10, eps_depth_divk = 5.4195, lambda_a = 6., lambda_r = 9., mass = 6.6464764e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 1, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH1")
- type(saftvrmie_data), parameter **miecx15** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "HE", m = 1., sigma = 2.549e-10, eps_depth_divk = 10.952, lambda_a = 6., lambda_r = 13., mass = 6.6464764e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 2, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH2")

- type(saftvrmie_data), parameter **miecx16** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "HE", m = 1., sigma = 3.1273e-10, eps_depth_divk = 4.1463, lambda_a = 6., lambda_r = 12., mass = 6.6464764e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH0-LJ")
- type(saftvrmie_data), parameter **miecx17** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "HE", m = 1., sigma = 2.8107e-10, eps_depth_divk = 6.6893, lambda_a = 6., lambda_r = 12., mass = 6.6464764e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 1, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH1-LJ")
- type(saftvrmie_data), parameter **miecx18** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "HE", m = 1., sigma = 2.5442e-10, eps_depth_divk = 10.393, lambda_a = 6., lambda_r = 12., mass = 6.6464764e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 2, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH2-LJ")
- type(saftvrmie_data), parameter **miecx19** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "HE", m = 1., sigma = 2.8525e-10, eps_depth_divk = 4.5764, lambda_a = 6., lambda_r = 8.3471, mass = 6.↩ 6464764e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "HYVA parameters with non-integer repulsive exponent", ref = "HYVA-FH0")
- type(saftvrmie_data), parameter **miecx20** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "HE", m = 1., sigma = 2.6615e-10, eps_depth_divk = 7.9191, lambda_a = 6., lambda_r = 10.455, mass = 6.↩ 6464764e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 1, bib_ref = "HYVA parameters with non-integer repulsive exponent", ref = "HYVA-FH1")
- type(saftvrmie_data), parameter **miecx21** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "HE", m = 1., sigma = 2.5682e-10, eps_depth_divk = 11.942, lambda_a = 6., lambda_r = 13.239, mass = 6.↩ 6464764e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 2, bib_ref = "HYVA parameters with non-integer repulsive exponent", ref = "HYVA-FH2")
- type(saftvrmie_data), parameter **miecx22** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "N2H4", m = 1.7839, sigma = 3.0012e-10, eps_depth_divk = 373., lambda_a = 6., lambda_r = 30.823, mass = 0.e+00, eps = 1400., beta = 9.9666E-28, assoc_scheme = assoc_scheme_4C, fh_order = 0, bib_ref = "Polishuk (2018) - 10.1016/j.molliq.2018.05.112", ref = "DEFAULT/Polishuk")
- type(saftvrmie_data), parameter **miecx23** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "H2", m = 1., sigma = 3.2574e-10, eps_depth_divk = 17.931, lambda_a = 6., lambda_r = 8., mass = 3.3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "DEFAULT/AASEN2019-FH0")
- type(saftvrmie_data), parameter **miecx24** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "H2", m = 1., sigma = 3.0243e-10, eps_depth_divk = 26.706, lambda_a = 6., lambda_r = 9., mass = 3.3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 1, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH1")
- type(saftvrmie_data), parameter **miecx25** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "H2", m = 1., sigma = 2.9195e-10, eps_depth_divk = 55.729, lambda_a = 6., lambda_r = 20., mass = 3.3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 2, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH2")
- type(saftvrmie_data), parameter **miecx26** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "H2", m = 1., sigma = 3.5333e-10, eps_depth_divk = 14.312, lambda_a = 6., lambda_r = 12., mass = 3.3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH0-LJ")
- type(saftvrmie_data), parameter **miecx27** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "H2", m = 1., sigma = 3.0225e-10, eps_depth_divk = 33.096, lambda_a = 6., lambda_r = 12., mass = 3.3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 1, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH1-LJ")
- type(saftvrmie_data), parameter **miecx28** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "H2", m = 1., sigma = 2.9324e-10, eps_depth_divk = 40.321, lambda_a = 6., lambda_r = 12., mass = 3.3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 2, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH2-LJ")
- type(saftvrmie_data), parameter **miecx29** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "H2", m = 1., sigma = 3.198e-10, eps_depth_divk = 14.492, lambda_a = 6., lambda_r = 6.4709, mass = 3.3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "HYVA parameters with non-integer repulsive exponent", ref = "HYVA-FH0")

- type(saftvrmie_data), parameter **miecx30** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "H2", m = 1., sigma = 3.0022e-10, eps_depth_divk = 28.349, lambda_a = 6., lambda_r = 9.5413, mass = 3.3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 1, bib_ref = "HYVA parameters with non-integer repulsive exponent", ref = "HYVA-FH1")
- type(saftvrmie_data), parameter **miecx31** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "H2", m = 1., sigma = 2.9209e-10, eps_depth_divk = 53.07, lambda_a = 6., lambda_r = 18.033, mass = 3.3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 2, bib_ref = "HYVA parameters with non-integer repulsive exponent", ref = "HYVA-FH2")
- type(saftvrmie_data), parameter **miecx32** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "H2", m = 1., sigma = 3.0459e-10, eps_depth_divk = 33.434, lambda_a = 6., lambda_r = 12., mass = 3.3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Trejos et al. 2013, doi: 10.1063/1.4829769", ref = "TREJOS2013")
- type(saftvrmie_data), parameter **miecx33** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "H2S", m = 1., sigma = 3.782e-10, eps_depth_divk = 243.28, lambda_a = 6., lambda_r = 31.311, mass = 5.659e-26, eps = 585.72, beta = 1.8804E-27, assoc_scheme = assoc_scheme_4C, fh_order = 0, bib_ref = "Dufal (2015) - 10.1080/00268976.2015.1029027", ref = "DEFAULT/Dufal2015")
- type(saftvrmie_data), parameter **miecx34** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "KR", m = 1., sigma = 3.64e-10, eps_depth_divk = 166.66, lambda_a = 6., lambda_r = 12., mass = 1.↩ 3914985275103401e-25, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Irma et. al (1999) ISSN : 1029-0435/0892-7022", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx35** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "LJF", m = 1., sigma = 3.e-10, eps_depth_divk = 100., lambda_a = 6., lambda_r = 12., mass = 0.e+00, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx36** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "C1", m = 1., sigma = 3.7412e-10, eps_depth_divk = 153.36, lambda_a = 6., lambda_r = 12.65, mass = 0.e+00, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Lafitte et al. 2013, doi: 10.1063/1.4819786", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx37** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "C1", m = 1., sigma = 3.752e-10, eps_depth_divk = 170.75, lambda_a = 6., lambda_r = 16.39, mass = 0.e+00, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "MÃüÂ¼ller's mystery parameters for methane and decane where SAFT-VR Mie fails", ref = "Muller")
- type(saftvrmie_data), parameter **miecx38** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "MEOH", m = 1.7989, sigma = 3.1425e-10, eps_depth_divk = 276.92, lambda_a = 6., lambda_r = 16.↩ 968, mass = 0.e+00, eps = 2156., beta = 2.2218E-28, assoc_scheme = assoc_scheme_3B, fh_order = 0, bib_ref = "Dufal (2015) - 10.1080/00268976.2015.1029027", ref = "DEFAULT/Dufal2015")
- type(saftvrmie_data), parameter **miecx39** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NE", m = 1., sigma = 2.8019e-10, eps_depth_divk = 29.875, lambda_a = 6., lambda_r = 9.6977, mass = 3.↩ 3509177e-26, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "DEFAULT/AASEN2019-FH0")
- type(saftvrmie_data), parameter **miecx40** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NE", m = 1., sigma = 2.7778e-10, eps_depth_divk = 37.501, lambda_a = 6., lambda_r = 13., mass = 3.3509177e-26, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 1, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH1")
- type(saftvrmie_data), parameter **miecx41** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NE", m = 1., sigma = 2.776e-10, eps_depth_divk = 37.716, lambda_a = 6., lambda_r = 13., mass = 3.3509177e-26, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 2, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH2")
- type(saftvrmie_data), parameter **miecx42** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NE", m = 1., sigma = 2.8066e-10, eps_depth_divk = 33.977, lambda_a = 6., lambda_r = 12., mass = 3.3509177e-26, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH0-LJ")
- type(saftvrmie_data), parameter **miecx43** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NE", m = 1., sigma = 2.776e-10, eps_depth_divk = 35.851, lambda_a = 6., lambda_r = 12., mass = 3.3509177e-26, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 1, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH1-LJ")
- type(saftvrmie_data), parameter **miecx44** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NE", m = 1., sigma = 2.7745e-10, eps_depth_divk = 36.024, lambda_a = 6., lambda_r = 12., mass = 3.3509177e-

26, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 2, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH2-LJ")

- type(saftvrmie_data), parameter **miecx45** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NE", m = 1., sigma = 2.8e-10, eps_depth_divk = 31.297, lambda_a = 6., lambda_r = 10.396, mass = 3.3509177e-26, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "HYVA parameters with non-integer repulsive exponent", ref = "HYVA-FH0")
- type(saftvrmie_data), parameter **miecx46** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NE", m = 1., sigma = 2.7765e-10, eps_depth_divk = 36.648, lambda_a = 6., lambda_r = 12.451, mass = 3.↩ 3509177e-26, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 1, bib_ref = "HYVA parameters with non-integer repulsive exponent", ref = "HYVA-FH1")
- type(saftvrmie_data), parameter **miecx47** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NE", m = 1., sigma = 2.7765e-10, eps_depth_divk = 38.039, lambda_a = 6., lambda_r = 13.187, mass = 3.↩ 3509177e-26, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 2, bib_ref = "HYVA parameters with non-integer repulsive exponent", ref = "HYVA-FH2")
- type(saftvrmie_data), parameter **miecx48** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NE", m = 1., sigma = 2.7867e-10, eps_depth_divk = 41.6523, lambda_a = 6., lambda_r = 16., mass = 3.3509177e-26, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 1, bib_ref = "Hammer 2022 not published", ref = "HAMMER-2022-NP")
- type(saftvrmie_data), parameter **miecx49** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "N2", m = 1., sigma = 3.656e-10, eps_depth_divk = 98.94, lambda_a = 6., lambda_r = 12.26, mass = 4.65173451e-26, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Ronald A. Aziz, J. Chem. Phys. 99, 4518 (1993), DOI : https://doi.org/10.1063/1.466051", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx50** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "O-H2", m = 1., sigma = 3.2571e-10, eps_depth_divk = 17.935, lambda_a = 6., lambda_r = 8., mass = 3.↩ 3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "DEFAULT/AASEN2019-FH0")
- type(saftvrmie_data), parameter **miecx51** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "O-H2", m = 1., sigma = 3.0239e-10, eps_depth_divk = 26.716, lambda_a = 6., lambda_r = 9., mass = 3.↩ 3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 1, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH1")
- type(saftvrmie_data), parameter **miecx52** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "O-H2", m = 1., sigma = 2.9191e-10, eps_depth_divk = 55.749, lambda_a = 6., lambda_r = 20., mass = 3.↩ 3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 2, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH2")
- type(saftvrmie_data), parameter **miecx53** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "O-H2", m = 1., sigma = 3.3604e-10, eps_depth_divk = 14.084, lambda_a = 6., lambda_r = 12., mass = 3.↩ 3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH0-LJ")
- type(saftvrmie_data), parameter **miecx54** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "O-H2", m = 1., sigma = 3.0222e-10, eps_depth_divk = 33.107, lambda_a = 6., lambda_r = 12., mass = 3.↩ 3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 1, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH1-LJ")
- type(saftvrmie_data), parameter **miecx55** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "O-H2", m = 1., sigma = 2.9319e-10, eps_depth_divk = 40.342, lambda_a = 6., lambda_r = 12., mass = 3.↩ 3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 2, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH2-LJ")
- type(saftvrmie_data), parameter **miecx56** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "O2", m = 1., sigma = 3.433e-10, eps_depth_divk = 113., lambda_a = 6., lambda_r = 12., mass = 5.31339291e-26, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Chen, Rex and Yuen, W., Oxidation of Metals, 73 (2009), DOI : 10.1007/s11085-009-9180-z", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx57** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "P-H2", m = 1., sigma = 3.2557e-10, eps_depth_divk = 17.849, lambda_a = 6., lambda_r = 8., mass = 3.↩ 3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "DEFAULT/AASEN2019-FH0")
- type(saftvrmie_data), parameter **miecx58** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "P-H2", m = 1., sigma = 3.0235e-10, eps_depth_divk = 26.586, lambda_a = 6., lambda_r = 9., mass = 3.↩ 3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 1, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH1")

- type(saftvrmie_data), parameter **miecx59** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "P-H2", m = 1., sigma = 2.9185e-10, eps_depth_divk = 55.519, lambda_a = 6., lambda_r = 20., mass = 3.↩ 3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 2, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH2")
- type(saftvrmie_data), parameter **miecx60** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "P-H2", m = 1., sigma = 3.8131e-10, eps_depth_divk = 13.197, lambda_a = 6., lambda_r = 12., mass = 3.↩ 3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH0-LJ")
- type(saftvrmie_data), parameter **miecx61** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "P-H2", m = 1., sigma = 3.0217e-10, eps_depth_divk = 32.955, lambda_a = 6., lambda_r = 12., mass = 3.↩ 3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 1, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH1-LJ")
- type(saftvrmie_data), parameter **miecx62** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "P-H2", m = 1., sigma = 2.9318e-10, eps_depth_divk = 40.152, lambda_a = 6., lambda_r = 12., mass = 3.↩ 3472e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 2, bib_ref = "Aasen 2019, doi: 10.1063/1.5111364", ref = "AASEN2019-FH2-LJ")
- type(saftvrmie_data), parameter **miecx63** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "C3", m = 1.6845, sigma = 3.9056e-10, eps_depth_divk = 239.89, lambda_a = 6., lambda_r = 13.006, mass = 0.e+00, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Lafitte et al. 2013, doi: 10.1063/1.4819786", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx64** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "H2O", m = 1., sigma = 3.0555e-10, eps_depth_divk = 418., lambda_a = 6., lambda_r = 35.823, mass = 2.992e-26, eps = 1600., beta = 4.9666E-28, assoc_scheme = assoc_scheme_4C, fh_order = 0, bib_ref = "Dufal (2015) - 10.1080/00268976.2015.1029027", ref = "DEFAULT/Dufal2015")
- type(saftvrmie_data), parameter **miecx65** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "H2O", m = 1.25656, sigma = 2.802422e-10, eps_depth_divk = 351.2321, lambda_a = 6., lambda_r = 25.↩ 12615, mass = 2.992e-26, eps = 1630.57, beta = 1.7762E-28, assoc_scheme = assoc_scheme_4C, fh_order = 0, bib_ref = "Unpublished water parameters (courtesy of Edward Graham)", ref = "Graham")
- type(saftvrmie_data), parameter **miecx66** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "XE", m = 1., sigma = 3.9011e-10, eps_depth_divk = 227.55, lambda_a = 6., lambda_r = 12., mass = 2.1801716e-22, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Jadran Vrabec, Jurgen Stoll, Hans Hasse, J. Phys. Chem. B 2001, 105, DOI: 10.1021/jp012542o", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx67** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NC4", m = 1.8514, sigma = 4.0887e-10, eps_depth_divk = 273.64, lambda_a = 6., lambda_r = 13.65, mass = 6.689e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Lafitte et al. 2013, doi: 10.1063/1.4819786", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx68** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NC10", m = 2.9976, sigma = 4.589e-10, eps_depth_divk = 400.79, lambda_a = 6., lambda_r = 18.885, mass = 6.689e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Lafitte et al. 2013, doi: 10.1063/1.4819786", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx69** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NC10", m = 3., sigma = 4.584e-10, eps_depth_divk = 415.19, lambda_a = 6., lambda_r = 20.92, mass = 6.689e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "MÃ¼ller's mystery parameters for methane and decane where SAFT-VR Mie fails", ref = "Muller")
- type(saftvrmie_data), parameter **miecx70** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NC22", m = 3.2519, sigma = 4.7484e-10, eps_depth_divk = 437.72, lambda_a = 6., lambda_r = 20.862, mass = 6.689e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Lafitte et al. 2013, doi: 10.1063/1.4819786", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx71** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NC20", m = 4.8794, sigma = 4.8788e-10, eps_depth_divk = 475.76, lambda_a = 6., lambda_r = 22.926, mass = 6.689e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Lafitte et al. 2013, doi: 10.1063/1.4819786", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx72** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NC7", m = 2.3949, sigma = 4.4282e-10, eps_depth_divk = 358.51, lambda_a = 6., lambda_r = 17.092, mass = 6.689e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Lafitte et al. 2013, doi: 10.1063/1.4819786", ref = "DEFAULT")

- type(saftvrmie_data), parameter **miecx73** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NC6", m = 2.1097, sigma = 4.423e-10, eps_depth_divk = 354.38, lambda_a = 6., lambda_r = 17.203, mass = 6.689e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Lafitte et al. 2013, doi: 10.1063/1.4819786", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx74** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NC9", m = 2.8099, sigma = 4.5334e-10, eps_depth_divk = 387.55, lambda_a = 6., lambda_r = 18.324, mass = 6.689e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Lafitte et al. 2013, doi: 10.1063/1.4819786", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx75** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NC8", m = 2.6253, sigma = 4.4696e-10, eps_depth_divk = 369.18, lambda_a = 6., lambda_r = 17.378, mass = 6.689e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Lafitte et al. 2013, doi: 10.1063/1.4819786", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx76** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NC15", m = 3.9325, sigma = 4.7738e-10, eps_depth_divk = 444.51, lambda_a = 6., lambda_r = 20.822, mass = 6.689e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Lafitte et al. 2013, doi: 10.1063/1.4819786", ref = "DEFAULT")
- type(saftvrmie_data), parameter **miecx77** = saftvrmie_data(eosidx = eosSAFT_VR_MIE, compName = "NC5", m = 1.9606, sigma = 4.2928e-10, eps_depth_divk = 321.94, lambda_a = 6., lambda_r = 15.847, mass = 6.689e-27, eps = 0., beta = 0.0000E+00, assoc_scheme = no_assoc, fh_order = 0, bib_ref = "Lafitte et al. 2013, doi: 10.1063/1.4819786", ref = "DEFAULT")
- integer, parameter **nmiemodels** = 77
- type(saftvrmie_data), dimension(nmiemodels), parameter **miearray** = (/ Miecx1,Miecx2,Miecx3,Miecx4,Miecx5, Miecx6,Miecx7,Miecx8,Miecx9,Miecx10, Miecx11,Miecx12,Miecx13,Miecx14,Miecx15, Miecx16,Miecx17,Miecx18,Miecx19,Mie Miecx21,Miecx22,Miecx23,Miecx24,Miecx25, Miecx26,Miecx27,Miecx28,Miecx29,Miecx30, Miecx31,Miecx32,Miecx33,Miecx3 Miecx36,Miecx37,Miecx38,Miecx39,Miecx40, Miecx41,Miecx42,Miecx43,Miecx44,Miecx45, Miecx46,Miecx47,Miecx48,Miecx4 Miecx51,Miecx52,Miecx53,Miecx54,Miecx55, Miecx56,Miecx57,Miecx58,Miecx59,Miecx60, Miecx61,Miecx62,Miecx63,Miecx6 Miecx66,Miecx67,Miecx68,Miecx69,Miecx70, Miecx71,Miecx72,Miecx73,Miecx74,Miecx75, Miecx76,Miecx77 /)
- type(miekijdata), parameter **svrm_kij_1** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "DEFAULT", bib_ref = "", uid1 = "He", uid2 = "Ne", kijvalue = 0.0425)
- type(miekijdata), parameter **svrm_kij_2** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "DEFAULT", bib_ref = "", uid1 = "C2", uid2 = "NC10", kijvalue = -0.0222)
- type(miekijdata), parameter **svrm_kij_3** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "DEFAULT", bib_ref = "", uid1 = "H2S", uid2 = "C2", kijvalue = 0.072)
- type(miekijdata), parameter **svrm_kij_4** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH1", bib_ref = "10.1063/1.5136079", uid1 = "He", uid2 = "Ne", kijvalue = -0.22)
- type(miekijdata), parameter **svrm_kij_5** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH1", bib_ref = "10.1063/1.5136079", uid1 = "H2", uid2 = "Ne", kijvalue = 0.105)
- type(miekijdata), parameter **svrm_kij_6** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH1", bib_ref = "10.1063/1.5136079", uid1 = "He", uid2 = "H2", kijvalue = 0.08)
- type(miekijdata), parameter **svrm_kij_7** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH1", bib_ref = "10.1063/1.5136079", uid1 = "D2", uid2 = "Ne", kijvalue = 0.13)
- type(miekijdata), parameter **svrm_kij_8** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH1", bib_ref = "10.1063/1.5136079", uid1 = "H2", uid2 = "D2", kijvalue = 0.)
- type(miekijdata), parameter **svrm_kij_9** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH1", bib_ref = "10.1063/1.5136079", uid1 = "He", uid2 = "D2", kijvalue = 0.)
- type(miekijdata), parameter **svrm_kij_10** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH2", bib_ref = "10.1063/1.5136079", uid1 = "He", uid2 = "Ne", kijvalue = -0.06)
- type(miekijdata), parameter **svrm_kij_11** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH2", bib_ref = "10.1063/1.5136079", uid1 = "H2", uid2 = "Ne", kijvalue = 0.105)
- type(miekijdata), parameter **svrm_kij_12** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH2", bib_ref = "10.1063/1.5136079", uid1 = "He", uid2 = "H2", kijvalue = 0.15)
- type(miekijdata), parameter **svrm_kij_13** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH2", bib_ref = "10.1063/1.5136079", uid1 = "D2", uid2 = "Ne", kijvalue = 0.14)
- type(miekijdata), parameter **svrm_kij_14** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH2", bib_ref = "10.1063/1.5136079", uid1 = "H2", uid2 = "D2", kijvalue = -0.04)

- type(miekijdata), parameter **svrm_kij_15** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH2", bib_ref = "10.1063/1.5136079", uid1 = "He", uid2 = "D2", kijvalue = 0.12)
- type(miekijdata), parameter **svrm_kij_16** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "HAMMER2022-FH1", bib_ref = "xxx", uid1 = "D2", uid2 = "Ne", kijvalue = 0.13)
- type(miekijdata), parameter **svrm_kij_17** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "HAMMER2022-NP", bib_ref = "Hammer 2022 not published", uid1 = "H2", uid2 = "Ne", kijvalue = 0.09)
- type(miekijdata), parameter **svrm_kij_18** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "Default/Dufal2015", bib_ref = "10.1080/00268976.2015.1029027", uid1 = "H2O", uid2 = "MEOH", kijvalue = 0.02)
- type(miekijdata), parameter **svrm_lij_1** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH1", bib_ref = "10.1063/1.5136079", uid1 = "He", uid2 = "Ne", kijvalue = 0.)
- type(miekijdata), parameter **svrm_lij_2** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH1", bib_ref = "10.1063/1.5136079", uid1 = "H2", uid2 = "Ne", kijvalue = 0.)
- type(miekijdata), parameter **svrm_lij_3** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH1", bib_ref = "10.1063/1.5136079", uid1 = "He", uid2 = "H2", kijvalue = -0.05)
- type(miekijdata), parameter **svrm_lij_4** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH1", bib_ref = "10.1063/1.5136079", uid1 = "D2", uid2 = "Ne", kijvalue = 0.)
- type(miekijdata), parameter **svrm_lij_5** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH1", bib_ref = "10.1063/1.5136079", uid1 = "H2", uid2 = "D2", kijvalue = 0.)
- type(miekijdata), parameter **svrm_lij_6** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH1", bib_ref = "10.1063/1.5136079", uid1 = "He", uid2 = "D2", kijvalue = -0.05)
- type(miekijdata), parameter **svrm_lij_7** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH2", bib_ref = "10.1063/1.5136079", uid1 = "He", uid2 = "Ne", kijvalue = 0.)
- type(miekijdata), parameter **svrm_lij_8** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH2", bib_ref = "10.1063/1.5136079", uid1 = "H2", uid2 = "Ne", kijvalue = 0.)
- type(miekijdata), parameter **svrm_lij_9** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH2", bib_ref = "10.1063/1.5136079", uid1 = "He", uid2 = "H2", kijvalue = -0.025)
- type(miekijdata), parameter **svrm_lij_10** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH2", bib_ref = "10.1063/1.5136079", uid1 = "D2", uid2 = "Ne", kijvalue = 0.)
- type(miekijdata), parameter **svrm_lij_11** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH2", bib_ref = "10.1063/1.5136079", uid1 = "H2", uid2 = "D2", kijvalue = 0.)
- type(miekijdata), parameter **svrm_lij_12** = Miekijdata(eosidx = eosSAFT_VR_MIE, ref = "AASEN2019-FH2", bib_ref = "10.1063/1.5136079", uid1 = "He", uid2 = "D2", kijvalue = -0.05)
- integer, parameter **miemaxkij** = 18
- type(miekijdata), dimension(miemaxkij), parameter **miekijdb** = (/ SVRM_KIJ_1,SVRM_KIJ_2,SVRM_↩ KIJ_3,SVRM_KIJ_4,SVRM_KIJ_5, SVRM_KIJ_6,SVRM_KIJ_7,SVRM_KIJ_8,SVRM_KIJ_9,SVRM_KIJ_10, SVRM_KIJ_11,SVRM_KIJ_12,SVRM_KIJ_13,SVRM_KIJ_14,SVRM_KIJ_15, SVRM_KIJ_16,SVRM_KIJ_↩ 17,SVRM_KIJ_18 /)
- integer, parameter **miemaxlij** = 12
- type(miekijdata), dimension(miemaxlij), parameter **mielijdb** = (/ SVRM_LIJ_1,SVRM_LIJ_2,SVRM_↩ LIJ_3,SVRM_LIJ_4,SVRM_LIJ_5, SVRM_LIJ_6,SVRM_LIJ_7,SVRM_LIJ_8,SVRM_LIJ_9,SVRM_LIJ_10, SVRM_LIJ_11,SVRM_LIJ_12 /)

5.49.1 Detailed Description

Automatically generated to file saftvrmie_datadb.f90 using utility python code pyUtils Time stamp: 2023-06-21T13↩ :07:27.307865.

5.50 satdensdatadb Module Reference

Automatically generated file satdensdatadb.f90 Time stamp: 2021-07-25T15:05:48.939509.

Data Types

- type satdensdata

Variables

- type(satdensdata), parameter **satdensgas_1** = satdensdata(ident = "AR", name = "ARGON", phase = "G", tr = 150.687, rhor = 13.40742965, correlation_id = 5, n_param = 4, n = (/ -0.29182D+01,0.97930D-01,-0.↩ 13721D+01, -0.22898D+01,0.0d0,0.0d0 /), t = (/ 0.72,1.25,0.32, 4.34,0.0d0,0.0d0 /))
- type(satdensdata), parameter **satdensliq_1** = satdensdata(ident = "AR", name = "ARGON", phase = "L", tr = 150.687, rhor = 13.40742965, correlation_id = 3, n_param = 4, n = (/ 1.5004264,-0.3138129,0.086461622, -0.041477525,0.0d0,0.0d0 /), t = (/ 0.334,0.6666666666666,2.3333333333333, 4.0,0.0d0,0.0d0 /))
- type(satdensdata), parameter **satdensgas_2** = satdensdata(ident = "CO", name = "CO", phase = "G", tr = 132.86, rhor = 10.85, correlation_id = 3, n_param = 6, n = (/ -0.25439D+01,-0.55601D+01,-0.85276D+01, -0.51163D+01,-0.17701D+02,-0.29858D+02 /), t = (/ 0.395,1.21,3.0, 3.5,6.0,8.0 /))
- type(satdensdata), parameter **satdensliq_2** = satdensdata(ident = "CO", name = "CO", phase = "L", tr = 132.86, rhor = 10.85, correlation_id = 1, n_param = 5, n = (/ 0.29570D+01,-0.42880D+01,0.87643D+01, -0.84001D+01,0.36372D+01,0.0d0 /), t = (/ 0.398,0.735,1.08, 1.5,1.9,0.0d0 /))
- type(satdensdata), parameter **satdensgas_3** = satdensdata(ident = "H2O", name = "WATER", phase = "G", tr = 647.096, rhor = 17.8737279956, correlation_id = 4, n_param = 6, n = (/ -2.03150240,-2.68302940,-5.↩ 38626492, -17.2991605,-44.7586581,-63.9201063 /), t = (/ 1.0,2.0,4.0, 9.0,18.5,35.5 /))
- type(satdensdata), parameter **satdensliq_3** = satdensdata(ident = "H2O", name = "WATER", phase = "L", tr = 647.096, rhor = 17.8737279956, correlation_id = 2, n_param = 6, n = (/ 1.99274064,1.09965342,-0.↩ 510839303, -1.75493479,-45.5170352,-6.74694450d5 /), t = (/ 1.,2.,5., 16.,43.,110. /))
- type(satdensdata), parameter **satdensgas_4** = satdensdata(ident = "CO2", name = "CO2", phase = "G", tr = 304.1282, rhor = 10.6249, correlation_id = 4, n_param = 5, n = (/ -1.7074879,-0.8227467,-4.6008549, -10.111178,-29.742252,0.0d0 /), t = (/ 1.02,1.5,3.0, 7.0,14.0,0.0d0 /))
- type(satdensdata), parameter **satdensliq_4** = satdensdata(ident = "CO2", name = "CO2", phase = "L", tr = 304.1282, rhor = 10.6249, correlation_id = 4, n_param = 4, n = (/ 1.92451080,-0.62385555,-0.32731127, 0.39245142,0.0d0,0.0d0 /), t = (/ 1.02,1.5,5.0, 5.5,0.0d0,0.0d0 /))
- type(satdensdata), parameter **satdensgas_5** = satdensdata(ident = "NC10", name = "DECANE", phase = "G", tr = 617.7, rhor = 1.64, correlation_id = 3, n_param = 6, n = (/ -0.50378D+01,-0.34694D+01,-0.↩ 15906D+02, -0.82894D+02,0.29336D+02,-0.10985D+03 /), t = (/ 0.4985,1.33,2.43, 5.44,5.8,11.0 /))
- type(satdensdata), parameter **satdensliq_5** = satdensdata(ident = "NC10", name = "DECANE", phase = "L", tr = 617.7, rhor = 1.64, correlation_id = 1, n_param = 5, n = (/ 0.92435D+01,-0.16288D+02,0.20445D+02, -0.17624D+02,0.73796D+01,0.0d0 /), t = (/ 0.535,0.74,1.0, 1.28,1.57,0.0d0 /))
- type(satdensdata), parameter **satdensgas_6** = satdensdata(ident = "O2", name = "OXYGEN", phase = "G", tr = 154.581, rhor = 13.63, correlation_id = 3, n_param = 6, n = (/ -0.22695D+01,-0.46578D+01,-0.99480D+01, -0.22845D+02,-0.45190D+02,-0.25101D+02 /), t = (/ 0.3785,1.07,2.7, 5.5,10.0,20.0 /))
- type(satdensdata), parameter **satdensliq_6** = satdensdata(ident = "O2", name = "OXYGEN", phase = "L", tr = 154.581, rhor = 13.63, correlation_id = 1, n_param = 5, n = (/ 0.16622D+01,0.76846D+00,-0.10041D+00, 0.20480D+00,0.11551D-01,0.0d0 /), t = (/ 0.345,0.74,1.2, 2.6,7.2,0.0d0 /))
- type(satdensdata), parameter **satdensgas_7** = satdensdata(ident = "NC8", name = "OCTANE", phase = "G", tr = 569.32, rhor = 2.056404, correlation_id = 3, n_param = 6, n = (/ -0.16556D+00,-0.59337D+01,-0.↩ 18915D+02, -0.36484D+03,0.72686D+03,-0.50392D+03 /), t = (/ 0.09,0.59,2.4, 7.0,8.0,9.0 /))
- type(satdensdata), parameter **satdensliq_7** = satdensdata(ident = "NC8", name = "OCTANE", phase = "L", tr = 569.32, rhor = 2.056404, correlation_id = 1, n_param = 5, n = (/ 0.56814D-01,0.38908D+02,-0.75923D+02, 0.59548D+02,-0.19651D+02,0.0d0 /), t = (/ 0.10,0.75,0.90, 1.10,1.25,0.0d0 /))
- type(satdensdata), parameter **satdensgas_8** = satdensdata(ident = "IC4", name = "ISOBUTAN", phase = "G", tr = 407.81, rhor = 3.879756788, correlation_id = 6, n_param = 4, n = (/ -2.12933323,-2.93790085,-0.↩ 89441086, -3.46343707,0.0d0,0.0d0 /), t = (/ 1.065,2.5,9.5, 13.0,0.0d0,0.0d0 /))
- type(satdensdata), parameter **satdensliq_8** = satdensdata(ident = "IC4", name = "ISOBUTAN", phase = "L", tr = 407.81, rhor = 3.879756788, correlation_id = 2, n_param = 4, n = (/ 2.04025104,0.850874089,-0.↩ 479052281, 0.348201252,0.0d0,0.0d0 /), t = (/ 1.065,3.0,4.0, 7.0,0.0d0,0.0d0 /))
- type(satdensdata), parameter **satdensgas_9** = satdensdata(ident = "HE", name = "HELIUM", phase = "G", tr = 5.1953, rhor = 17.3837, correlation_id = 3, n_param = 5, n = (/ -1.5789,-10.749,17.711, -15.413,-14.↩ 352,0.0d0 /), t = (/ 0.333,1.5,2.1, 2.7,9.0,0.0d0 /))
- type(satdensdata), parameter **satdensliq_9** = satdensdata(ident = "HE", name = "HELIUM", phase = "L", tr = 5.1953, rhor = 17.3837, correlation_id = 1, n_param = 5, n = (/ 1.0929,1.6584,-3.6477, 2.7440,-2.3859,0.0d0 /), t = (/ 0.286,1.2,2.0, 2.8,6.5,0.0d0 /))

- type(satdensdata), parameter **satdensgas_10** = satdensdata(ident = "H2", name = "HYDROGEN", phase = "G", tr = 33.145, rhor = 15.508, correlation_id = 3, n_param = 6, n = (/ -0.29962D+01,-0.16724D+02,0.↩ 15819D+02, -0.16852D+02,0.34586D+02,-0.53754D+02 /), t = (/ 0.466,2.,2.4, 4.,7.,8. /))
- type(satdensdata), parameter **satdensliq_10** = satdensdata(ident = "H2", name = "HYDROGEN", phase = "L", tr = 33.145, rhor = 15.508, correlation_id = 1, n_param = 5, n = (/ 0.15456D+02,-0.41720D+02,0.↩ 50276D+02, -0.27947D+02,0.56718D+01,0.0d0 /), t = (/ 0.62,0.83,1.05, 1.3,1.6,0.0d0 /))
- type(satdensdata), parameter **satdensgas_11** = satdensdata(ident = "NC5", name = "PENTANE", phase = "G", tr = 469.7, rhor = 3.2155776, correlation_id = 3, n_param = 6, n = (/ -0.29389D+01,-0.62784D+01,-0.↩ 19941D+02, -0.16709D+02,-0.36543D+02,-0.12799D+03 /), t = (/ 0.4,1.18,3.2, 6.6,7.0,15.0 /))
- type(satdensdata), parameter **satdensliq_11** = satdensdata(ident = "NC5", name = "PENTANE", phase = "L", tr = 469.7, rhor = 3.2155776, correlation_id = 1, n_param = 5, n = (/ 0.10178D+01,0.42703D+00,0.↩ 11334D+01, 0.41518D+00,-0.47950D-01,0.0d0 /), t = (/ 0.27,0.44,0.6, 4.0,5.0,0.0d0 /))
- type(satdensdata), parameter **satdensgas_12** = satdensdata(ident = "NC4", name = "BUTANE", phase = "G", tr = 425.125, rhor = 3.922769613, correlation_id = 3, n_param = 5, n = (/ -0.27390D+01,-0.57347D+01,-0.16408D+02, -0.46986D+02,-0.10090D+03,0.0d0 /), t = (/ 0.391,1.14,3.0, 6.5,14.0,0.0d0 /))
- type(satdensdata), parameter **satdensliq_12** = satdensdata(ident = "NC4", name = "BUTANE", phase = "L", tr = 425.125, rhor = 3.922769613, correlation_id = 1, n_param = 4, n = (/ 0.52341D+01,-0.62011D+01,0.↩ 36063D+01, 0.22137D+00,0.0d0,0.0d0 /), t = (/ 0.44,0.60,0.76, 5.00,0.0d0,0.0d0 /))
- type(satdensdata), parameter **satdensgas_13** = satdensdata(ident = "N2", name = "NITROGEN", phase = "G", tr = 126.192, rhor = 11.1839, correlation_id = 6, n_param = 5, n = (/ -0.170127164E+1,-0.↩ 370402649E+1,0.129859383E+1, -0.561424977E+0,-0.268505381E+1,0.0d0 /), t = (/ 1.02,2.5,3.5, 6.5,14.↩ 0,0.0d0 /))
- type(satdensdata), parameter **satdensliq_13** = satdensdata(ident = "N2", name = "NITROGEN", phase = "L", tr = 126.192, rhor = 11.1839, correlation_id = 4, n_param = 4, n = (/ 0.148654237E+1,-0.280476066E+0,0.↩ 894143085E-1, -0.119879866E+0,0.0d0,0.0d0 /), t = (/ 0.9882,2.,8., 17.5,0.0d0,0.0d0 /))
- type(satdensdata), parameter **satdensgas_14** = satdensdata(ident = "NC7", name = "HEPTANE", phase = "G", tr = 540.13, rhor = 2.315323, correlation_id = 3, n_param = 6, n = (/ -0.24571D+00,-0.63004D+01,-0.↩ 19144D+02, -0.96970D+02,0.21643D+03,-0.27953D+03 /), t = (/ 0.097,0.646,2.56, 6.6,9.3,10.7 /))
- type(satdensdata), parameter **satdensliq_14** = satdensdata(ident = "NC7", name = "HEPTANE", phase = "L", tr = 540.13, rhor = 2.315323, correlation_id = 1, n_param = 5, n = (/ -0.26395D+01,0.21806D+02,-0.↩ 28896D+02, 0.12609D+02,0.40749D+00,0.0d0 /), t = (/ 0.322,0.504,0.651, 0.816,6.4,0.0d0 /))
- type(satdensdata), parameter **satdensgas_15** = satdensdata(ident = "C2", name = "ETHANE", phase = "G", tr = 305.322, rhor = 6.856886685, correlation_id = 6, n_param = 6, n = (/ -1.89879145,-3.65459262,0.↩ 850562745, 0.363965487,-1.50005943,-2.26690389 /), t = (/ 1.038,2.5,3.0, 6.0,9.0,15.0 /))
- type(satdensdata), parameter **satdensliq_15** = satdensdata(ident = "C2", name = "ETHANE", phase = "L", tr = 305.322, rhor = 6.856886685, correlation_id = 4, n_param = 4, n = (/ 1.56138026,-0.381552776,0.↩ 0785372040, 0.0370315089,0.0d0,0.0d0 /), t = (/ 0.987,2.0,4.0, 9.5,0.0d0,0.0d0 /))
- type(satdensdata), parameter **satdensgas_16** = satdensdata(ident = "H2S", name = "H2S", phase = "G", tr = 373.1, rhor = 10.19, correlation_id = 3, n_param = 4, n = (/ -3.9156,-7.7093,-19.543, -49.418,0.0d0,0.0d0 /), t = (/ 0.49,1.69,4.00, 8.00,0.0d0,0.0d0 /))
- type(satdensdata), parameter **satdensliq_16** = satdensdata(ident = "H2S", name = "H2S", phase = "L", tr = 373.1, rhor = 10.19, correlation_id = 2, n_param = 3, n = (/ 11.833,-17.019,7.8047, 0.0d0,0.0d0,0.0d0 /), t = (/ 1.63,1.95,2.30, 0.0d0,0.0d0,0.0d0 /))
- type(satdensdata), parameter **satdensgas_17** = satdensdata(ident = "IC5", name = "IPENTANE", phase = "G", tr = 460.356, rhor = 3.271, correlation_id = 3, n_param = 6, n = (/ -0.38825D+02,0.79040D+02,-0.↩ 48791D+02, -0.21603D+02,-0.57218D+02,-0.15164D+03 /), t = (/ 0.565,0.66,0.77, 3.25,7.3,16.6 /))
- type(satdensdata), parameter **satdensliq_17** = satdensdata(ident = "IC5", name = "IPENTANE", phase = "L", tr = 460.356, rhor = 3.271, correlation_id = 1, n_param = 5, n = (/ 0.18367D+02,-0.30283D+02,0.13557D+02, -0.90533D+00,0.20927D+01,0.0d0 /), t = (/ 1.21,1.41,1.65, 0.09,0.164,0.0d0 /))
- type(satdensdata), parameter **satdensgas_18** = satdensdata(ident = "C1", name = "METHANE", phase = "G", tr = 190.564, rhor = 10.139128, correlation_id = 4, n_param = 6, n = (/ -1.8802840,-2.8526531,-3.↩ 0006480, -5.2511690,-13.191859,-37.553961 /), t = (/ 1.062,2.5,4.5, 7.5,12.5,23.5 /))
- type(satdensdata), parameter **satdensliq_18** = satdensdata(ident = "C1", name = "METHANE", phase = "L", tr = 190.564, rhor = 10.139128, correlation_id = 3, n_param = 3, n = (/ 1.9906389,-0.78756197,0.036976723, 0.0d0,0.0d0,0.0d0 /), t = (/ 0.354,0.5,2.5, 0.0d0,0.0d0,0.0d0 /))
- type(satdensdata), parameter **satdensgas_19** = satdensdata(ident = "NC6", name = "HEXANE", phase = "G", tr = 507.82, rhor = 2.7058779, correlation_id = 3, n_param = 6, n = (/ -0.13309D+00,-0.50653D+01,-0.↩ 11602D+02, -0.28530D+02,-0.51731D+02,-0.13482D+03 /), t = (/ 0.107,0.553,2.006, 4.46,8.0,16.0 /))

- type(satdensdata), parameter **satdensliq_19** = satdensdata(ident = "NC6", name = "HEXANE", phase = "L", tr = 507.82, rhor = 2.7058779, correlation_id = 1, n_param = 3, n = (/ 0.14686D+03,-0.26585D+03,0.↩ 12200D+03, 0.0d0,0.0d0,0.0d0 /), t = (/ 0.75,0.81,0.88, 0.0d0,0.0d0,0.0d0 /))
- type(satdensdata), parameter **satdensgas_20** = satdensdata(ident = "C3", name = "PROPANE", phase = "G", tr = 369.89, rhor = 5.0, correlation_id = 3, n_param = 6, n = (/ -2.4887,-5.1069,-12.174, -30.495,-52.↩ 192,-134.89 /), t = (/ 0.3785,1.07,2.7, 5.5,10.0,20.0 /))
- type(satdensdata), parameter **satdensliq_20** = satdensdata(ident = "C3", name = "PROPANE", phase = "L", tr = 369.89, rhor = 5.0, correlation_id = 1, n_param = 4, n = (/ 1.82205,0.65802,0.21109, 0.083973,0.0d0,0.↩ 0d0 /), t = (/ 0.345,0.74,2.6, 7.2,0.0d0,0.0d0 /))
- type(satdensdata), parameter **satdensgas_21** = satdensdata(ident = "NC9", name = "NONANE", phase = "G", tr = 594.55, rhor = 1.81, correlation_id = 3, n_param = 5, n = (/ -0.33199D+01,-0.23900D+01,-0.↩ 15307D+02, -0.51788D+02,-0.11133D+03,0.0d0 /), t = (/ 0.461,0.666,2.12, 5.1,11.0,0.0d0 /))
- type(satdensdata), parameter **satdensliq_21** = satdensdata(ident = "NC9", name = "NONANE", phase = "L", tr = 594.55, rhor = 1.81, correlation_id = 1, n_param = 5, n = (/ -0.43785D+00,0.37240D+01,-0.23029D+01, 0.18270D+01,0.38664D+00,0.0d0 /), t = (/ 0.116,0.32,0.54, 0.8,3.5,0.0d0 /))
- integer, parameter **maxsatdens** = 42
- type(satdensdata), dimension(maxsatdens), parameter **satdensdb** = (/ satdensgas_1,satdensliq↩ _1,satdensgas_2,satdensliq_2,satdensgas_3,satdensliq_3, satdensgas_4,satdensliq_4,satdensgas↩ _5,satdensliq_5,satdensgas_6,satdensliq_6, satdensgas_7,satdensliq_7,satdensgas_8,satdensliq_↩ 8,satdensgas_9,satdensliq_9, satdensgas_10,satdensliq_10,satdensgas_11,satdensliq_11,satdensgas_↩ 12,satdensliq_12, satdensgas_13,satdensliq_13,satdensgas_14,satdensliq_14,satdensgas_15,satdensliq↩ _15, satdensgas_16,satdensliq_16,satdensgas_17,satdensliq_17,satdensgas_18,satdensliq_18, satdensgas↩ _19,satdensliq_19,satdensgas_20,satdensliq_20,satdensgas_21,satdensliq_21 /)

5.50.1 Detailed Description

Automatically generated file satdensdatadb.f90 Time stamp: 2021-07-25T15:05:48.939509.

5.51 saturation_point_locators Module Reference

Functionality for locating points on the saturation curve based on auxiliary properties such as specific volume or entropy. The isentropeEnvelopeCross routines should eventually be moved from saturation to saturation_point_locators.

Functions/Subroutines

- subroutine, public bracketsolveforpropertysingle (ic, propflag, propspec, phase, t1, p1, dpdt1, ts, ps, ierr)

  Bracket solver for finding the exact point on saturation line.
- subroutine, public sat_points_based_on_prop (z, t0, p0, x0, y0, n_grid, propflag, prop_grid, t_grid, p_grid, phase_grid, wi_grid, n_grid_found, ds_override, phase_in, ierr_out, sgn_in, spec_in, tmin, tmax, pmin, pmax, normal_grid, sequential_mode)

  Locate the points on the phase envelope corresponding to the property values in prop_grid. The grid must be sorted according to whether one starts on the dew curve or the bubble curve.
- subroutine, public iso_cross_saturation_line (ts, ps, t, p, x, y, z, beta, prop_spec, prop_specid, phase, ierr)

  Iso-line crossing of saturation line.

Variables

- integer, parameter, public **locate_from_entropy** = 1
- integer, parameter, public **locate_from_lnvol** = 2
- integer, parameter, public **locate_from_enthalpy** = 3
- integer, parameter, public **locate_from_temperature** = 4
- integer, parameter, public **locate_from_pressure** = 5
- integer, parameter, public **locate_from_joule_thompson** = 6

5.51.1 Detailed Description

Functionality for locating points on the saturation curve based on auxiliary properties such as specific volume or entropy. The isentropeEnvelopeCross routines should eventually be moved from saturation to saturation_point_locators.

Author

Ailo 2015-12

MH 2018-06

5.51.2 Function/Subroutine Documentation

5.51.2.1 bracketsolveforpropertysingle()

```
subroutine, public saturation_point_locators::bracketsolveforpropertysingle (
             integer, intent(in) ic,
             integer, intent(in) propflag,
             real, intent(in) propspec,
             integer, intent(in) phase,
             real, intent(in) t1,
             real, intent(in) p1,
             real, intent(in) dpdt1,
             real, intent(inout) ts,
             real, intent(inout) ps,
             integer, intent(out), optional ierr )
```

Bracket solver for finding the exact point on saturation line.
What property is is determined by propflag.

Author

MH 2019-05

5.51.2.2 iso_cross_saturation_line()

```
subroutine, public saturation_point_locators::iso_cross_saturation_line (
             real, intent(in) ts,
             real, intent(in) ps,
             real, intent(inout) t,
             real, intent(inout) p,
             real, dimension(nc), intent(in) x,
             real, dimension(nc), intent(in) y,
             real, dimension(nc), intent(in) z,
             real, intent(inout) beta,
             real, intent(in) prop_spec,
             character, intent(in) prop_specid,
             integer, intent(out) phase,
             integer, intent(out) ierr )
```

Iso-line crossing of saturation line.

Author

MH, 2017-06

Parameters

in	*ts*	Single phase temperature (K)
in	*ps*	Single phase pressure (Pa)
in,out	*t*	Two-phase temperature (K)
in,out	*p*	Two-phase pressure (Pa)

Parameters

in,out	*beta*	Two-phase gas phase fraction (mol/mol)
in	*x*	Liquid composition
in	*y*	Gas composition
in	*prop_specid*	"h", "s", "t", "p"
out	*phase*	Single phase at envelope crossing
out	*ierr*	Error indicator, zero on success

5.51.2.3 sat_points_based_on_prop()

```
subroutine, public saturation_point_locators::sat_points_based_on_prop (
             real, dimension(nc), intent(in) z,
             real, intent(in) t0,
             real, intent(in) p0,
             real, dimension(nc), intent(in) x0,
             real, dimension(nc), intent(in) y0,
             integer, intent(in) n_grid,
             integer, intent(in) propflag,
             real, dimension(n_grid), intent(inout) prop_grid,
             real, dimension(n_grid), intent(out) t_grid,
             real, dimension(n_grid), intent(out) p_grid,
             integer, dimension(n_grid), intent(out) phase_grid,
             real, dimension(nc,n_grid), intent(out) wi_grid,
             integer, intent(out) n_grid_found,
             real, intent(in), optional ds_override,
             integer, intent(in), optional phase_in,
             integer, intent(out), optional ierr_out,
             real, intent(in), optional sgn_in,
             integer, intent(in), optional spec_in,
             real, intent(in), optional tmin,
             real, intent(in), optional tmax,
             real, intent(in), optional pmin,
             real, intent(in), optional pmax,
             real, dimension(2,n_grid), intent(out), optional normal_grid,
             logical, intent(in), optional sequential_mode )
```

Locate the points on the phase envelope corresponding to the property values in prop_grid. The grid must be sorted according to whether one starts on the dew curve or the bubble curve.
The routine starts at the dew point at given initial pressure, and then traverses the phase envelope until it has bracketed a grid value; it then locates the exact saturation point corresponding to this value, before continuing.
This routine can perhaps be generalized to subsume the routines envelopePlot and isentropeEnvelopeCross.
To do: test the routine when starting from bubble curve.

Author

Ailo 2015-12

Morten 2018-06

Parameters

in	*z*	total composition
in	*p0*	init. point has pressure p0
in	*y0*	init. liq./vap. compo. guess
in	*n_grid*	number of grid points
in	*propflag*	specified property 1:s, 2:lnv

Parameters

in,out	*prop_grid*	property grid (non sequential mode may reorder array)
in	*ds_override*	step length along envelope
in	*phase_in*	start search from this phase
out	*ierr_out*	error flag; nonzero if error
in	*sgn_in*	Override search direction
in	*spec_in*	Override initial search direction
in	*tmax*	Temperature limits (K)
in	*pmax*	Pressure limits (Pa)
in	*sequential_mode*	Default sequential
out	*t_grid*	t at the grid points
out	*p_grid*	p at the grid points
out	*phase_grid*	incumbent phase at grid points
out	*wi_grid*	Incipient phase at boundary
out	*n_grid_found*	Number of grid points found
out	*normal_grid*	Normal vector (from tangent) pointing into two-phase region

5.52 solid_correlation_datadb Module Reference

Automatically generated to file solid_correlation_datadb.f90 using utility python code pyUtils Time stamp: 2023-04-28T20:02:12.169534.

Data Types

- type solid_correlation_data

  This data structure stores parameters for sublimation and melting line correlations.

Variables

- type(solid_correlation_data), parameter **melt1** = solid_correlation_data(compName = "NH3", correlation = "ML-1", triple_temperature = 195.495, minimum_temperature = 0., maximum_temperature = 10000., reducing_pressure = 1000000., reducing_temperature = 195.49453, n_coeff = 1, n_coeff_1 = 0, n_coeff_2 = 0, n_coeff_3 = 1, coeff = (/2.533125000000e+03,0.000000000000e+00,0.000000000000e+00, 0.000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/1.000000,0.000000,0.000000, 0.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1063/1.555579", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt2** = solid_correlation_data(compName = "AR", correlation = "ML-1", triple_temperature = 83.8058, minimum_temperature = 0., maximum_temperature = 700., reducing_pressure = 68891., reducing_temperature = 83.8058, n_coeff = 3, n_coeff_1 = 1, n_coeff_2 = 0, n_coeff_3 = 0, coeff = (/1.000000000000e+00,-7.476266510000e+03,9.959061250000e+03, 0.000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,1.050000,1.275000, 0.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1063/1.556037", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt3** = solid_correlation_data(compName = "CO2", correlation = "ML-1", triple_temperature = 216.592, minimum_temperature = 0., maximum_temperature = 1100., reducing_pressure = 517950., reducing_temperature = 216.592, n_coeff = 3, n_coeff_1 = 0, n_coeff_2 = 3, n_coeff_3 = 0, coeff = (/1.000000000000e+00,1.955539000000e+03,2.055459300000e+03, 0.000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,1.000000,2.000000, 0.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1063/1.555991", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt4** = solid_correlation_data(compName = "CO", correlation = "ML-1", triple_temperature = 68.16, minimum_temperature = 0., maximum_temperature = 1000., reducing_pressure = 1000000., reducing_temperature = 1., n_coeff = 2, n_coeff_1 = 2, n_coeff_2 = 0, n_coeff_3 = 0, coeff = (/-1.429410000000e+02,1.956080000000e-02,0.000000000000e+00, 0.000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,2.107470,0.000000, 0.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1016/0021-9614(82)90044-1", ref = "DEFAULT")

- type(solid_correlation_data), parameter **melt5** = solid_correlation_data(compName = "CYCLOHEX", correlation = "ML-1", triple_temperature = 279.86, minimum_temperature = 0., maximum_temperature = 700., reducing_pressure = 5348.7, reducing_temperature = 279.86, n_coeff = 3, n_coeff_1 = 2, n_coeff_2 = 1, n_coeff_3 = 0, coeff = (/1.000000000000e+00,7.500000000000e+01,1.020000000000e+05, 0.000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,2.000000,1.000000, 0.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1063/1.4900538", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt6** = solid_correlation_data(compName = "C2", correlation = "ML-1", triple_temperature = 90.368, minimum_temperature = 0., maximum_temperature = 2000., reducing_pressure = 1.1421, reducing_temperature = 90.368, n_coeff = 3, n_coeff_1 = 1, n_coeff_2 = 1, n_coeff_3 = 0, coeff = (/1.000000000000e+00,2.236263150000e+08,1.052623740000e+08, 0.000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,1.000000,2.550000, 0.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1063/1.1859286", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt7** = solid_correlation_data(compName = "HE", correlation = "ML-1", triple_temperature = 2.1768, minimum_temperature = 0., maximum_temperature = 1500., reducing_pressure = 1000000., reducing_temperature = 1., n_coeff = 2, n_coeff_1 = 2, n_coeff_2 = 0, n_coeff_3 = 0, coeff = (/-1.745583700000e+00,1.697979300000e+00,0.000000000000e+00, 0.000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,1.555414,0.000000, 0.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1007/978-1-4613-0639-9_174", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt8** = solid_correlation_data(compName = "H2", correlation = "ML-1", triple_temperature = 13.957, minimum_temperature = 0., maximum_temperature = 400., reducing_pressure = 7357.8, reducing_temperature = 13.957, n_coeff = 3, n_coeff_1 = 1, n_coeff_2 = 2, n_coeff_3 = 0, coeff = (/1.000000000000e+00,5.626300000000e+03,2.717200000000e+03, 0.000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,1.000000,1.830000, 0.000000,0.000000,0.000000/), bib_ref = "", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt9** = solid_correlation_data(compName = "IC4", correlation = "ML-1", triple_temperature = 113.73, minimum_temperature = 0., maximum_temperature = 575., reducing_pressure = 0.022891, reducing_temperature = 113.73, n_coeff = 2, n_coeff_1 = 2, n_coeff_2 = 0, n_coeff_3 = 0, coeff = (/-1.953637129000e+09,1.953637130000e+09,0.000000000000e+00, 0.000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,6.120000,0.000000, 0.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1063/1.1901687", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt10** = solid_correlation_data(compName = "IC5", correlation = "ML-1", triple_temperature = 112.65, minimum_temperature = 0., maximum_temperature = 2000., reducing_pressure = 0.00008952, reducing_temperature = 112.65, n_coeff = 2, n_coeff_1 = 2, n_coeff_2 = 0, n_coeff_3 = 0, coeff = (/-7.127700000000e+12,7.127700000001e+12,0.000000000000e+00, 0.000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,1.563000,0.000000, 0.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1063/1.1725068", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt11** = solid_correlation_data(compName = "KR", correlation = "ML-1", triple_temperature = 115.775, minimum_temperature = 0., maximum_temperature = 800., reducing_pressure = 101325., reducing_temperature = 1., n_coeff = 2, n_coeff_1 = 2, n_coeff_2 = 0, n_coeff_3 = 0, coeff = (/-2.345921000000e+03,1.080476685000e+00,0.000000000000e+00, 0.000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,1.616984,0.000000, 0.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1016/0031-8914(62)90096-4", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt12** = solid_correlation_data(compName = "C1", correlation = "ML-1", triple_temperature = 90.6941, minimum_temperature = 0., maximum_temperature = 625., reducing_pressure = 11696., reducing_temperature = 90.6941, n_coeff = 3, n_coeff_1 = 1, n_coeff_2 = 0, n_coeff_3 = 0, coeff = (/1.000000000000e+00,2.475680000000e+04,-7.366020000000e+03, 0.000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,1.850000,2.100000, 0.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1063/1.555898", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt13** = solid_correlation_data(compName = "MEOH", correlation = "ML-1", triple_temperature = 175.61, minimum_temperature = 0., maximum_temperature = 620., reducing_pressure = 0.187, reducing_temperature = 175.61, n_coeff = 4, n_coeff_1 = 1, n_coeff_2 = 3, n_coeff_3 = 0, coeff = (/1.000000000000e+00,5.320770000000e+09,4.524780000000e+09, 3.888861000000e+10,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,1.000000,1.500000, 4.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1016/j.fluid.2013.03.024", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt14** = solid_correlation_data(compName = "NE", correlation = "ML-1", triple_temperature = 24.556, minimum_temperature = 0., maximum_temperature = 10000., reducing_pressure = 43368.14, reducing_temperature = 24.556, n_coeff = 2, n_coeff_1 = 1, n_coeff_2 = 1, n_coeff_3 = 0, coeff = (/1.000000000000e+00,4.437000000000e+03,0.000000000000e+00, 0.

000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,1.330000,0.↩
000000, 0.000000,0.000000,0.000000/), bib_ref = "Lemmon, E.W. (2010)", ref = "DEFAULT")

- type(solid_correlation_data), parameter **melt15** = solid_correlation_data(compName = "N2", correlation = "ML-1", triple_temperature = 63.151, minimum_temperature = 0., maximum_temperature = 2000., reducing↩
_pressure = 12519.8, reducing_temperature = 63.151, n_coeff = 2, n_coeff_1 = 1, n_coeff_2 = 0, n_coeff_3 = 0, coeff = (/1.000000000000e+00,1.279861000000e+04,0.000000000000e+00, 0.000000000000e+00,0.↩
000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,1.789630,0.000000, 0.000000,0.↩
000000,0.000000/), bib_ref = "DOI: 10.1063/1.1349047", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt16** = solid_correlation_data(compName = "O2", correlation = "ML-2", triple_temperature = 54.361, minimum_temperature = 0., maximum_temperature = 300., reducing↩
_pressure = 146.277, reducing_temperature = 54.361, n_coeff = 4, n_coeff_1 = 0, n_coeff_2 = 4, n_coeff_3 = 0, coeff = (/-3.246353900000e+01,1.427801100000e+02,-1.470234100000e+02, 5.200120000000e+01,0.↩
000000000000e+00,0.000000000000e+00/), exponents = (/0.062500,0.125000,0.187500, 0.250000,0.↩
000000,0.000000/), bib_ref = "DOI: 10.1016/0378-3812(85)87016-3", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt17** = solid_correlation_data(compName = "P-H2", correlation = "MP-1", triple_temperature = 13.8, minimum_temperature = 0., maximum_temperature = 400., reducing_↩
pressure = 1000000., reducing_temperature = 1., n_coeff = 4, n_coeff_1 = 4, n_coeff_2 = 0, n_coeff_3 = 0, coeff = (/-2.652891150000e+01,2.485785960000e-01,-2.128233930000e+01, 1.257466430000e-01,0.↩
000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,1.764739,0.000000, 1.955000,0.↩
000000,0.000000/), bib_ref = "ISBN: 088318415X, https://srd.nist.gov/JPCRD/jpcrdS1Vol11.pdf", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt18** = solid_correlation_data(compName = "C3", correlation = "ML-1", triple_temperature = 85.525, minimum_temperature = 0., maximum_temperature = 2000., reducing_pressure = 0.00017205, reducing_temperature = 85.525, n_coeff = 2, n_coeff_1 = 2, n_coeff_2 = 0, n_coeff_3 = 0, coeff = (/-4.230000000000e+12,4.230000000001e+12,0.000000000000e+00, 0.↩
000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,1.283000,0.↩
000000, 0.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1063/1.1725068", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt19** = solid_correlation_data(compName = "PRLN", correlation = "ML-1", triple_temperature = 87.953, minimum_temperature = 0., maximum_temperature = 2000., reducing_pressure = 0.0007471, reducing_temperature = 87.953, n_coeff = 2, n_coeff_1 = 2, n_coeff↩
_2 = 0, n_coeff_3 = 0, coeff = (/-6.593000000000e+09,6.593000001000e+09,0.000000000000e+00, 0.↩
000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,2.821000,0.↩
000000, 0.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1063/1.1725068", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt20** = solid_correlation_data(compName = "F6S", correlation = "ML-1", triple_temperature = 223.555, minimum_temperature = 0., maximum_temperature = 650., reducing_pressure = 231429., reducing_temperature = 223.555, n_coeff = 2, n_coeff_1 = 1, n_coeff↩
_2 = 0, n_coeff_3 = 0, coeff = (/1.000000000000e+00,9.666031480000e+02,0.000000000000e+00, 0.↩
000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,1.555000,0.↩
000000, 0.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1063/1.5005537", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt21** = solid_correlation_data(compName = "XE", correlation = "ML-1", triple_temperature = 161.405, minimum_temperature = 0., maximum_temperature = 1300., reducing_pressure = 101325., reducing_temperature = 1., n_coeff = 2, n_coeff_1 = 2, n_coeff_↩
2 = 0, n_coeff_3 = 0, coeff = (/-2.575072800000e+03,7.983277028000e-01,0.000000000000e+00, 0.↩
000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,1.589165,0.↩
000000, 0.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1016/0031-8914(62)90096-4", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt22** = solid_correlation_data(compName = "NC4", correlation = "ML-1", triple_temperature = 134.895, minimum_temperature = 0., maximum_temperature = 575., reducing_pressure = 0.66566, reducing_temperature = 134.895, n_coeff = 2, n_coeff_1 = 2, n_coeff_↩
2 = 0, n_coeff_3 = 0, coeff = (/-5.585582354000e+08,5.585582364000e+08,0.000000000000e+00, 0.↩
000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,2.206000,0.↩
000000, 0.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1063/1.1901687", ref = "DEFAULT")
- type(solid_correlation_data), parameter **melt23** = solid_correlation_data(compName = "NC5", correlation = "ML-1", triple_temperature = 143.47, minimum_temperature = 0., maximum_temperature = 2000., reducing_pressure = 0.076322, reducing_temperature = 143.47, n_coeff = 2, n_coeff_1 = 2, n_coeff_↩
2 = 0, n_coeff_3 = 0, coeff = (/-8.647500000000e+09,8.647500001000e+09,0.000000000000e+00, 0.↩
000000000000e+00,0.000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,1.649000,0.↩
000000, 0.000000,0.000000,0.000000/), bib_ref = "DOI: 10.1063/1.1725068", ref = "DEFAULT")

- type(solid_correlation_data), parameter **subl1** = solid_correlation_data(compName = "NH3", correlation = "SL-2", triple_temperature = 195.49, minimum_temperature = 0., maximum_temperature = 0., reducing_↩ pressure = 1000000., reducing_temperature = 1., n_coeff = 5, n_coeff_1 = 5, n_coeff_2 = 0, n_coeff_3 = 0, coeff = (/1.363932000000e+01,-3.537000000000e+03,-3.310000000000e+04, 1.742000000000e+06,-2.↩ 995000000000e+07,0.000000000000e+00/), exponents = (/0.000000,-1.000000,-2.000000, -3.000000,-4.↩ 000000,0.000000/), bib_ref = "DOI: 10.1063/5.0128269", ref = "DEFAULT")
- type(solid_correlation_data), parameter **subl2** = solid_correlation_data(compName = "AR", correlation = "SL-3", triple_temperature = 83.8058, minimum_temperature = 0., maximum_temperature = 0., reducing_↩ pressure = 68891., reducing_temperature = 83.8058, n_coeff = 1, n_coeff_1 = 0, n_coeff_2 = 1, n_coeff_3 = 0, coeff = (/-1.113070000000e+01,0.000000000000e+00,0.000000000000e+00, 0.000000000000e+00,0.↩ 000000000000e+00,0.000000000000e+00/), exponents = (/1.000000,0.000000,0.000000, 0.000000,0.↩ 000000,0.000000/), bib_ref = "Lemmon, E.W. (2002)", ref = "DEFAULT")
- type(solid_correlation_data), parameter **subl3** = solid_correlation_data(compName = "CO2", correlation = "SL-3", triple_temperature = 216.592, minimum_temperature = 0., maximum_temperature = 0., reducing_↩ pressure = 517950., reducing_temperature = 216.592, n_coeff = 3, n_coeff_1 = 0, n_coeff_2 = 3, n_coeff_3 = 0, coeff = (/-1.474084600000e+01,2.432701500000e+00,-5.306177800000e+00, 0.000000000000e+00,0.↩ 000000000000e+00,0.000000000000e+00/), exponents = (/1.000000,1.900000,2.900000, 0.000000,0.↩ 000000,0.000000/), bib_ref = "DOI: 10.1063/1.555991", ref = "DEFAULT")
- type(solid_correlation_data), parameter **subl4** = solid_correlation_data(compName = "CO", correlation = "SL-2", triple_temperature = 68.16, minimum_temperature = 61.55, maximum_temperature = 0., reducing↩ _pressure = 1000000., reducing_temperature = 1., n_coeff = 4, n_coeff_1 = 4, n_coeff_2 = 0, n_coeff_3 = 0, coeff = (/7.945240000000e+00,-7.481510000000e+02,-5.843300000000e+03, 3.938500000000e+04,0.↩ 000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,-1.000000,-2.000000, -3.000000,0.↩ 000000,0.000000/), bib_ref = "DOI: 10.1007/978-1-4613-9856-1_76", ref = "DEFAULT")
- type(solid_correlation_data), parameter **subl5** = solid_correlation_data(compName = "C2", correlation = "SL-2", triple_temperature = 90.368, minimum_temperature = 20., maximum_temperature = 0., reducing_↩ pressure = 1000000., reducing_temperature = 1., n_coeff = 6, n_coeff_1 = 6, n_coeff_2 = 0, n_coeff_3 = 0, coeff = (/1.281110000000e+01,-2.207460000000e+03,-2.411300000000e+04, 7.744400000000e+05,-1.↩ 161500000000e+07,6.763300000000e+07/), exponents = (/0.000000,-1.000000,-2.000000, -3.000000,-4.↩ 000000,-5.000000/), bib_ref = "DOI: 10.1007/978-1-4613-9856-1_76", ref = "DEFAULT")
- type(solid_correlation_data), parameter **subl6** = solid_correlation_data(compName = "H2", correlation = "SL-3", triple_temperature = 13.957, minimum_temperature = 0., maximum_temperature = 0., reducing↩ _pressure = 7700., reducing_temperature = 13.957, n_coeff = 1, n_coeff_1 = 0, n_coeff_2 = 1, n_coeff_3 = 0, coeff = (/-8.065000000000e+00,0.000000000000e+00,0.000000000000e+00, 0.000000000000e+00,0.↩ 000000000000e+00,0.000000000000e+00/), exponents = (/0.930000,0.000000,0.000000, 0.000000,0.↩ 000000,0.000000/), bib_ref = "Lemmon, E.W. (2003)", ref = "DEFAULT")
- type(solid_correlation_data), parameter **subl7** = solid_correlation_data(compName = "H2S", correlation = "S2", triple_temperature = 187.7, minimum_temperature = 0., maximum_temperature = 0., reducing_pressure = 1000000., reducing_temperature = 187.7, n_coeff = 5, n_coeff_1 = 5, n_coeff_2 = 0, n_coeff_3 = 0, coeff = (/6.624700000000e+00,-7.260000000000e+02,-3.504000000000e+05, 2.724000000000e+07,-8.↩ 582000000000e+08,0.000000000000e+00/), exponents = (/0.000000,-1.000000,-2.000000, -3.000000,-4.↩ 000000,0.000000/), bib_ref = "DOI: 10.1016/j.pss.2009.09.011", ref = "DEFAULT")
- type(solid_correlation_data), parameter **subl8** = solid_correlation_data(compName = "KR", correlation = "SL-3", triple_temperature = 115.775, minimum_temperature = 0., maximum_temperature = 0., reducing_↩ pressure = 73197., reducing_temperature = 115.775, n_coeff = 1, n_coeff_1 = 0, n_coeff_2 = 1, n_coeff_3 = 0, coeff = (/-1.156160000000e+01,0.000000000000e+00,0.000000000000e+00, 0.000000000000e+00,0.↩ 000000000000e+00,0.000000000000e+00/), exponents = (/1.000000,0.000000,0.000000, 0.000000,0.↩ 000000,0.000000/), bib_ref = "Lemmon, E.W. (2002)", ref = "DEFAULT")
- type(solid_correlation_data), parameter **subl9** = solid_correlation_data(compName = "C1", correlation = "SL-3", triple_temperature = 90.6941, minimum_temperature = 0., maximum_temperature = 0., reducing_↩ pressure = 11696., reducing_temperature = 90.6941, n_coeff = 1, n_coeff_1 = 0, n_coeff_2 = 1, n_coeff_3 = 0, coeff = (/-1.284000000000e+01,0.000000000000e+00,0.000000000000e+00, 0.000000000000e+00,0.↩ 000000000000e+00,0.000000000000e+00/), exponents = (/1.000000,0.000000,0.000000, 0.000000,0.↩ 000000,0.000000/), bib_ref = "Lemmon, E.W. (2002)", ref = "DEFAULT")
- type(solid_correlation_data), parameter **subl10** = solid_correlation_data(compName = "NE", correlation = "SL-2", triple_temperature = 24.556, minimum_temperature = 7., maximum_temperature = 0., reducing↩ _pressure = 1000000., reducing_temperature = 1., n_coeff = 5, n_coeff_1 = 5, n_coeff_2 = 0, n_coeff_3 =

0, coeff = (/8.307100000000e+00,-3.085550000000e+02,9.860200000000e+02, -9.069300000000e+03,3.↩
514200000000e+04,0.000000000000e+00/), exponents = (/0.000000,-1.000000,-2.000000, -3.000000,-4.↩
000000,0.000000/), bib_ref = "DOI: 10.1007/978-1-4613-9856-1_76", ref = "DEFAULT")

- type(solid_correlation_data), parameter **subl11** = solid_correlation_data(compName = "N2", correlation = "SL-3", triple_temperature = 63.151, minimum_temperature = 0., maximum_temperature = 0., reducing_↩ pressure = 12523., reducing_temperature = 63.151, n_coeff = 1, n_coeff_1 = 0, n_coeff_2 = 1, n_coeff_3 = 0, coeff = (/-1.308869200000e+01,0.000000000000e+00,0.000000000000e+00, 0.000000000000e+00,0.↩ 000000000000e+00,0.000000000000e+00/), exponents = (/1.000000,0.000000,0.000000, 0.000000,0.↩ 000000,0.000000/), bib_ref = "Lemmon, E.W. (1999)", ref = "DEFAULT")
- type(solid_correlation_data), parameter **subl12** = solid_correlation_data(compName = "N2O", correlation = "SL-2", triple_temperature = 182.33, minimum_temperature = 0., maximum_temperature = 0., reducing↩ _pressure = 1000000., reducing_temperature = 1., n_coeff = 4, n_coeff_1 = 4, n_coeff_2 = 0, n_coeff_3 = 0, coeff = (/1.379787540000e+01,-3.181209790000e+03,6.345151470000e+04, -4.189965370000e+06,0.↩ 000000000000e+00,0.000000000000e+00/), exponents = (/0.000000,-1.000000,-2.000000, -3.000000,0.↩ 000000,0.000000/), bib_ref = "Bell, I.H. (2018)", ref = "DEFAULT")
- type(solid_correlation_data), parameter **subl13** = solid_correlation_data(compName = "O2", correlation = "SL-3", triple_temperature = 54.361, minimum_temperature = 0., maximum_temperature = 0., reducing_↩ pressure = 146.28, reducing_temperature = 54.361, n_coeff = 1, n_coeff_1 = 0, n_coeff_2 = 1, n_coeff_3 = 0, coeff = (/-2.071400000000e+01,0.000000000000e+00,0.000000000000e+00, 0.000000000000e+00,0.↩ 000000000000e+00,0.000000000000e+00/), exponents = (/1.060000,0.000000,0.000000, 0.000000,0.↩ 000000,0.000000/), bib_ref = "Lemmon, E.W. (2003)", ref = "DEFAULT")
- type(solid_correlation_data), parameter **subl14** = solid_correlation_data(compName = "P-H2", correlation = "SL-2", triple_temperature = 13.8, minimum_temperature = 0., maximum_temperature = 0., reducing↩ _pressure = 1000000., reducing_temperature = 1., n_coeff = 6, n_coeff_1 = 6, n_coeff_2 = 0, n_coeff_3 = 0, coeff = (/4.782880000000e+00,-1.485636000000e+02,2.323210000000e+02, -5.602070000000e+02,6.↩ 641260000000e+02,-2.890600000000e+02/), exponents = (/0.000000,-1.000000,-2.000000, -3.000000,-4.↩ 000000,-5.000000/), bib_ref = "DOI: 10.1007/978-1-4613-9856-1_76", ref = "DEFAULT")
- type(solid_correlation_data), parameter **subl15** = solid_correlation_data(compName = "F6S", correlation = "SL-2", triple_temperature = 223.555, minimum_temperature = 0., maximum_temperature = 0., reducing_↩ pressure = 231429., reducing_temperature = 223.555, n_coeff = 2, n_coeff_1 = 2, n_coeff_2 = 0, n_coeff_3 = 0, coeff = (/-1.169421410000e+01,1.169421410000e+01,0.000000000000e+00, 0.000000000000e+00,0.↩ 000000000000e+00,0.000000000000e+00/), exponents = (/-1.070000,0.000000,0.000000, 0.000000,0.↩ 000000,0.000000/), bib_ref = "DOI: 10.1063/1.3037344", ref = "DEFAULT")
- type(solid_correlation_data), parameter **subl16** = solid_correlation_data(compName = "XE", correlation = "SL-3", triple_temperature = 161.405, minimum_temperature = 0., maximum_temperature = 0., reducing_↩ pressure = 81750., reducing_temperature = 161.405, n_coeff = 2, n_coeff_1 = 0, n_coeff_2 = 2, n_coeff_3 = 0, coeff = (/-1.390000000000e+01,1.400000000000e+01,0.000000000000e+00, 0.000000000000e+00,0.↩ 000000000000e+00,0.000000000000e+00/), exponents = (/1.060000,3.100000,0.000000, 0.000000,0.↩ 000000,0.000000/), bib_ref = "Lemmon, E.W. (2003)", ref = "DEFAULT")
- integer, parameter **n_melting_curves** = 23
- type(solid_correlation_data), dimension(n_melting_curves), parameter **melting_corr_array** = (/ MELT1,MELT2,MELT3,MELT4,
MELT6,MELT7,MELT8,MELT9,MELT10, MELT11,MELT12,MELT13,MELT14,MELT15, MELT16,MELT17,MELT18,MELT19,MELT
MELT21,MELT22,MELT23 /)
- integer, parameter **n_sublimation_curves** = 16
- type(solid_correlation_data), dimension(n_sublimation_curves), parameter **sublimation_corr_array** = (/ SUBL1,SUBL2,SUBL3,SUBL4,SUBL5, SUBL6,SUBL7,SUBL8,SUBL9,SUBL10, SUBL11,SUBL12,SUBL13,SUBL14,SUBL15, SUBL16 /)

5.52.1 Detailed Description

Automatically generated to file solid_correlation_datadb.f90 using utility python code pyUtils Time stamp: 2023-04-28T20:02:12.169534.

5.53 solideos Module Reference

Interface to solid equations of state. Currently only a Gibbs based equations of state for CO2 are supported.

Functions/Subroutines

- subroutine, public solid_init (comp)

  Initialize solid model for given component.
- subroutine, public solid_thermo (t, p, z, lnfug, lnfugt, lnfugp)

  Calculate solid fugasities etc. given composition, temperature and pressure.
- subroutine, public solid_enthalpy (t, p, z, h, dhdt, dhdp)

  Calculate solid enthalpy given composition, temperature and pressure Unit: J/mol.
- subroutine, public solid_entropy (t, p, z, s, dsdt, dsdp)

  Calculate solid entropy given composition, temperature and pressure Unit: J/mol/K.
- subroutine, public solid_specificvolume (t, p, z, v, dvdt, dvdp)

  Calculate solid specific volume given composition, temperature and pressure Unit: m3/mol.
- real function, public solidforming (t, p, j, z, nd, beta, ifugac, lnfugs)

  Calculate phase fraction of solid given current fugacities. If result is positive, the solid will be a stable phase.
- logical function, public isformingsolid (nd, t, p, z, beta, xx, phasevec, betasol)

  Return logical flag that is true when a solid phase is stable.
- real function, public solidfraction (j, z, nd, beta, ifugac, fugs)

  Calculate phase fraction of solid given current fugacities.
- subroutine, public initice ()

  Init water ice model Option to use Ice model without checking for ice in multiphase flash.
- subroutine, public initdryice ()

  Init dry ice model Option to use dry ice model without checking for dry ice in multiphase flash.

Variables

- integer, public **co2gibbsmodel**
- integer, public **h2ogibbsmodel**
- integer, dimension(:), allocatable, public **solidcomp**
- integer, public **nsolid** = 0

5.53.1 Detailed Description

Interface to solid equations of state. Currently only a Gibbs based equations of state for CO2 are supported.

Author

MH, 2013-03-05.

5.53.2 Function/Subroutine Documentation

5.53.2.1 initdryice()

`subroutine, public solideos::initdryice`

Init dry ice model Option to use dry ice model without checking for dry ice in multiphase flash.

Author

MH, 2015-11

5.53.2.2 initice()

`subroutine, public solideos::initice`

Init water ice model Option to use Ice model without checking for ice in multiphase flash.

Author

MH, 2015-11

5.53.2.3 isformingsolid()

```
logical function, public solideos::isformingsolid (
            integer, intent(in) nd,
            real, intent(inout) t,
            real, intent(inout) p,
            real, dimension(nc), intent(in) z,
            real, dimension(nph), intent(inout) beta,
            real, dimension(nph,nc), intent(inout) xx,
            integer, dimension(nph), intent(in) phasevec,
            real, intent(out), optional betasol )
```

Return logical flag that is true when a solid phase is stable.

Author

MH, 2016, 03

Parameters

in,out	*beta*	Phase molar fractions [-]
in	*z*	Overall molar compozition [-]
in,out	*xx*	Phase molar compozition [-]
in,out	*t*	Temperature [K]
in,out	*p*	Pressure [Pa]
in	*phasevec*	Phase identifiers
in	*nd*	Number of phases

5.53.2.4 solid_enthalpy()

```
subroutine, public solideos::solid_enthalpy (
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(1:nc), intent(in) z,
            real, intent(out) h,
            real, intent(out), optional dhdt,
            real, intent(out), optional dhdp )
```

Calculate solid enthalpy given composition, temperature and pressure Unit: J/mol.

Author

MH, 2013-03-06

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*z*	Compozition
out	*h*	J/mol - Specifc enthalpy
out	*dhdt*	J/mol/K - Specifc enthalpy differential wrpt. temperature
out	*dhdp*	J/mol/Pa - Specifc enthalpy differential wrpt. pressure

5.53.2.5 solid_entropy()

```
subroutine, public solideos::solid_entropy (
```

```
real, intent(in) t,
real, intent(in) p,
real, dimension(1:nc), intent(in) z,
real, intent(out) s,
real, intent(out), optional dsdt,
real, intent(out), optional dsdp )
```

Calculate solid entropy given composition, temperature and pressure Unit: J/mol/K.

Author

MH, 2013-03-06

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*z*	Compozition
out	*s*	J/mol/K - Specifc entropy
out	*dsdt*	J/mol/K2 - Specifc entropy differential wrpt. temperature
out	*dsdp*	J/mol/K/Pa - Specifc entropy differential wrpt. pressure

5.53.2.6 solid_init()

```
subroutine, public solideos::solid_init (
            character(len=*), intent(in) comp )
```

Initialize solid model for given component.

Author

MH, 2013-03-05

Parameters

in	*comp*	Component name string

5.53.2.7 solid_specificvolume()

```
subroutine, public solideos::solid_specificvolume (
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(1:nc), intent(in) z,
            real, intent(out) v,
            real, intent(out), optional dvdt,
            real, intent(out), optional dvdp )
```

Calculate solid specific volume given composition, temperature and pressure Unit: m3/mol.

Author

MH, 2013-03-06

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*z*	Compozition

Parameters

out	*v*	m3/mol - Specifc volume
out	*dvdt*	m3/mol/K - Specifc volume differential wrpt. temperature
out	*dvdp*	m3/mol/Pa - Specifc volume differential wrpt. pressure

5.53.2.8 solid_thermo()

```
subroutine, public solideos::solid_thermo (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(1:nc), intent(in) z,
             real, intent(out) lnfug,
             real, intent(out), optional lnfugt,
             real, intent(out), optional lnfugp )
```

Calculate solid fugasities etc. given composition, temperature and pressure.

Author

MH, 2013-03-05

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*z*	Compozition
out	*lnfug*	Logarithm of solid fugasity coefficient
out	*lnfugt*	1/K - Logarithm of solid fugasity coefficient differential wrpt. temperature
out	*lnfugp*	1/Pa - Logarithm of solid fugasity coefficient differential wrpt. pressure

5.53.2.9 solidforming()

```
real function, public solideos::solidforming (
             real, intent(in) t,
             real, intent(in) p,
             integer, intent(in) j,
             real, dimension(nc), intent(in) z,
             integer, intent(in) nd,
             real, dimension(nd), intent(in) beta,
             real, dimension(nd,nc), intent(in) ifugac,
             real, intent(out) lnfugs )
```

Calculate phase fraction of solid given current fugacities. If result is positive, the solid will be a stable phase.

Author

MH, 2013-03-07

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*j*	Solid component index
in	*nd*	Number of phases
in	*z*	Overall compozition

Parameters

in	*beta*	Phase fractions
in	*ifugac*	Inverse fugasity coefficient of exsisting phases
out	*lnfugs*	- Logarithm of solid fugasity coefficient

Returns

Solid phase fraction

5.53.2.10 solidfraction()

```
real function, public solideos::solidfraction (
             integer, intent(in) j,
             real, dimension(nc), intent(in) z,
             integer, intent(in) nd,
             real, dimension(nd), intent(in) beta,
             real, dimension(nd,nc), intent(in) ifugac,
             real, intent(in) fugs )
```

Calculate phase fraction of solid given current fugacities.

Author

MH, 2013-08-21

Parameters

in	*j*	Solid component index
in	*nd*	Number of phases
in	*z*	Overall compozition
in	*beta*	Phase fractions
in	*ifugac*	Inverse fugasity coefficient of exsisting phases
in	*fugs*	- Solid fugasity coefficient

Returns

Solid phase fraction

5.54 speed_of_sound Module Reference

Calculate speed of sound for full (P,T, chemical potential) equilibrium.

Functions/Subroutines

- real function, public singlephasespeedofsound (t, p, z, phase)

  Calculate speed-of-sound for single-phase flow.
- real function, public twophasespeedofsound (nph, t, p, z, beta, x, phase)

  Calculate speed-of-sound for liqid-liquid or gas-liquid mixtures.
- real function, public solidspeedofsound (t, p, z, betaf, xf, xs, phase)

  Calculate speed-of-sound for liqid-solid or gas-solid mixtures.
- real function, public sound_velocity_2ph (t, p, x, y, z, betav, betal, phase, ph)

  Calculate speed of sound for single phase or gas-liquid. Alternative interface to sound_velocity.
- real function, public speed_of_sound_tv (t, v, n)

  Calculate speed of sound for single phase given temperature, volume and mol numbers.

5.54.1 Detailed Description

Calculate speed of sound for full (P,T, chemical potential) equilibrium.

5.54.2 Function/Subroutine Documentation

5.54.2.1 singlephasespeedofsound()

```
real function, public speed_of_sound::singlephasespeedofsound (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             integer, intent(in) phase )
```

Calculate speed-of-sound for single-phase flow.

Author

KYL, 2013-10

Parameters

in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*z*	Overall molar compozition [-]
in	*phase*	Phase spec

5.54.2.2 solidspeedofsound()

```
real function, public speed_of_sound::solidspeedofsound (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             real, intent(in) betaf,
             real, dimension(nc), intent(in) xf,
             real, dimension(nc), intent(in) xs,
             integer, intent(in) phase )
```

Calculate speed-of-sound for liqid-solid or gas-solid mixtures.
Equation set considered:

$$f(1) = \ln\left(x^F(js)\varphi^F(js)\right) - \ln\left(\varphi^S\right)$$
$$f(2) = \frac{S_{\text{spec}} - S_{\text{mix}}}{R}$$
$$f(3) = \frac{P\left(v_{\text{mix}} - v_{\text{spec}}\right)}{RT}$$

Variable vector:

$$X^{\intercal} = \left[n^F(js), \ln T, \ln P\right]$$

Author

MH, 2013-09

Parameters

in	*betaf*	Fluid phase molar fraction [-]
in	*z*	Overall molar compozition [-]

Parameters

in	*xf*	Fluid molar compozition [-]
in	*xs*	Solid molar compozition [-]
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*phase*	Fluid phase integer

5.54.2.3 sound_velocity_2ph()

```
real function, public speed_of_sound::sound_velocity_2ph (
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) x,
            real, dimension(nc), intent(in) y,
            real, dimension(nc), intent(in) z,
            real, intent(in) betav,
            real, intent(in) betal,
            integer, intent(in) phase,
            integer, dimension(2), intent(in), optional ph )
```

Calculate speed of sound for single phase or gas-liquid. Alternative interface to sound_velocity.

Author

MH, 2014-05

Parameters

in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*z*	Overall molar compozition [-]
in	*betav*	Vapor molar fraction [-]
in	*betal*	Liquid molar fraction [-]
in	*x*	Liquid molar compozition [-]
in	*y*	Vapor molar compozition [-]
in	*phase*	Phase spec
in	*ph*	Override phase integers

Returns

Speed of sound [m/s]

5.54.2.4 speed_of_sound_tv()

```
real function, public speed_of_sound::speed_of_sound_tv (
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(nc), intent(in) n )
```

Calculate speed of sound for single phase given temperature, volume and mol numbers.

Author

MH, 2019-06

Parameters

in	*t*	Temperature [K]
in	*v*	Pressure [m3]
in	*n*	Mol numbers [mol]

Returns

Speed of sound [m/s]

5.54.2.5 twophasespeedofsound()

```
real function, public speed_of_sound::twophasespeedofsound (
             integer, intent(in) nph,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             real, dimension(nph), intent(in) beta,
             real, dimension(nph,nc), intent(in) x,
             integer, dimension(nph), intent(in) phase )
```

Calculate speed-of-sound for liqid-liquid or gas-liquid mixtures.

Author

MH, 2013-09

Parameters

in	*nph*	Maximum number of phases
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*z*	Overall molar compozition [-]
in	*beta*	Phase molar fraction [-]
in	*x*	Phase molar compozition [-]
in	*phase*	Phase spec

5.55 spinodal Module Reference

Methods for mapping spinodal.

Functions/Subroutines

- subroutine, public initial_stab_limit_point (p, z, v, t, phase, ierr, tmin)

  Given pressure find initial point (T,v) for mapping stability line.
- subroutine, public map_stability_limit (p0, z, tmin, tl, pl, vl, nl, ierr, dlnv_override, tliq_start)

  Map limit of stable phases.
- real function, public rhomax_pr (x)

  Calculate maximum density according to the Peng-Robinson EoS. Equals the inverse covolume. PR is preferred over SRK since it gives higher max.
- real function, public rho_of_meta_extremum (t, x, phase, rho_init_in)

  Computes the density at the first local pressure extremum for a general EoS. If no such extremum exists (i.e. a monotone rho-P curve), return a negative density.

- subroutine, public locate_spinodal_prop_pure_fluid (propflag, z, propspec, phase, p0, ts, vs, ps, found_↩ crossing, ierr, tliq_start)

  Locate property - spinodal curve intersect.
- subroutine, public locate_spinodal_prop_min_max_pure_fluid (propflag, z, p0, ts_min, vs_min, ps_min, ts↩ _max, vs_max, ps_max, ierr)

  Locate extrema on spinodal curve.
- subroutine, public tv_meta_ps (p, s, n, t, v, ierr)

  Solve for stable or meta-stable state given entropy and pressure.
- subroutine, public map_meta_isentrope (p0, s, n, pmin, nmax, t, v, p, ierr)

  Map single phase isentrope in meta-stable region.

Variables

- integer, parameter, public **nmax** = 1000
- integer, parameter, public **spin_locate_from_entropy** = 1
- integer, parameter, public **spin_locate_from_volume** = 2
- integer, parameter, public **spin_locate_from_enthalpy** = 3
- integer, parameter, public **spin_locate_from_temperature** = 4
- integer, parameter, public **spin_locate_from_pressure** = 5
- integer, parameter, public **spin_locate_from_energy** = 6

5.55.1 Detailed Description

Methods for mapping spinodal.

5.55.2 Function/Subroutine Documentation

5.55.2.1 initial_stab_limit_point()

```
subroutine, public spinodal::initial_stab_limit_point (
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             real, intent(out) v,
             real, intent(out) t,
             integer, intent(in) phase,
             integer, intent(out) ierr,
             real, intent(in), optional tmin )
```

Given pressure find initial point (T,v) for mapping stability line.

Author

MH, 2016-02

Parameters

in	*z*	Overall compozition
in	*p*	Pressure [Pa]
in	*phase*	Where to look for initial point
out	*t*	Temperature [K]
out	*v*	Specific volume [m3/mol]
out	*ierr*	Error flag

5.55.2.2 locate_spinodal_prop_min_max_pure_fluid()

```
subroutine, public spinodal::locate_spinodal_prop_min_max_pure_fluid (
```

```
            integer, intent(in) propflag,
            real, dimension(nc), intent(in) z,
            real, intent(in) p0,
            real, intent(out) ts_min,
            real, intent(out) vs_min,
            real, intent(out) ps_min,
            real, intent(out) ts_max,
            real, intent(out) vs_max,
            real, intent(out) ps_max,
            integer, intent(out) ierr )
```

Locate extrema on spinodal curve.

Author

MH, 2021-02

5.55.2.3 locate_spinodal_prop_pure_fluid()

```
subroutine, public spinodal::locate_spinodal_prop_pure_fluid (
            integer, intent(in) propflag,
            real, dimension(nc), intent(in) z,
            real, intent(in) propspec,
            integer, intent(in) phase,
            real, intent(in) p0,
            real, intent(out) ts,
            real, intent(out) vs,
            real, intent(out) ps,
            logical, intent(out) found_crossing,
            integer, intent(out) ierr,
            real, intent(in), optional tliq_start )
```

Locate property - spinodal curve intersect.

Author

MH, 2020-01

5.55.2.4 map_meta_isentrope()

```
subroutine, public spinodal::map_meta_isentrope (
            real, intent(in) p0,
            real, intent(in) s,
            real, dimension(nc), intent(in) n,
            real, intent(in) pmin,
            integer, intent(in) nmax,
            real, dimension(nmax), intent(out) t,
            real, dimension(nmax), intent(out) v,
            real, dimension(nmax), intent(out) p,
            integer, intent(out) ierr )
```

Map single phase isentrope in meta-stable region.

Author

MH, 2021-02

5.55.2.5 map_stability_limit()

```
subroutine, public spinodal::map_stability_limit (
            real, intent(in) p0,
            real, dimension(nc), intent(in) z,
```

```
            real, intent(in) tmin,
            real, dimension(nmax), intent(out) tl,
            real, dimension(nmax), intent(out) pl,
            real, dimension(nmax), intent(out) vl,
            integer, intent(out) nl,
            integer, intent(out) ierr,
            real, intent(in), optional dlnv_override,
            real, intent(in), optional tliq_start )
```

Map limit of stable phases.

Author

MH, 2015-11

Parameters

in	*z*	Overall compozition
in	*p0*	Pressure first point on line [Pa]
in	*tmin*	Stop mapping if T $<$ Tmin [K]
out	*tl*	Line temperature [K]
out	*vl*	Line specific volume [m3/mol]
out	*pl*	Line pressure [Pa]
out	*ierr*	Error flag
out	*nl*	Actual number of points on curve
in	*dlnv_override*	Volume step override [m3/mol]

5.55.2.6 rho_of_meta_extremum()

```
real function, public spinodal::rho_of_meta_extremum (
            real, intent(in) t,
            real, dimension(nce), intent(in) x,
            integer, intent(in) phase,
            real, intent(in), optional rho_init_in )
```

Computes the density at the first local pressure extremum for a general EoS. If no such extremum exists (i.e. a monotone rho-P curve), return a negative density.

Todo : May need more sophisticated method of choosing initial liquid rho.

: May need more robust handling of overshoots. Now we just "hope".

: May need to check that we have converged to the *correct* extremum.

Author

Ailo 2016-01

Parameters

in	*t*	Temperature [K]
in	*x*	Composition
in	*phase*	Phase flag; VAPPH or LIQPH
in	*rho_init_in*	Override initial rho if desired

Returns

[mol/m^3]

5.55.2.7 rhomax_pr()

```
real function, public spinodal::rhomax_pr (
            real, dimension(nce) x )
```

Calculate maximum density according to the Peng-Robinson EoS. Equals the inverse covolume. PR is preferred over SRK since it gives higher max.
There is no guarantee that this is the maximum for an EoS other than PR.

Author

Ailo 2016-01

Parameters

x	Composition (needn't be normalized)

Returns

Maximum density [mol/m^3]

5.55.2.8 tv_meta_ps()

```
subroutine, public spinodal::tv_meta_ps (
             real, intent(in) p,
             real, intent(in) s,
             real, dimension(nc), intent(in) n,
             real, intent(inout) t,
             real, intent(inout) v,
             integer, intent(out) ierr )
```

Solve for stable or meta-stable state given entropy and pressure.

Author

MH, 2021-02

5.56 stability Module Reference

Minimize reduced tangent plane distance.

Functions/Subroutines

- subroutine, public checkvlestability (t, p, z, isstable, wsol, new_phase)

  Check if a feed is stable against vapor-liquid split.
- real function, public stabcalc (t, p, z, phase, fugz, fug, wsol, pretermlim)

  Calculate minimum reduced tangent plane distance.
- real function, public stabcalcw (nd, k, t, p, xx, w, phase, fugz, fugw, pretermlim)

  Calculate minimum reduced tangent plane distance.
- real function, public tpd_fun (w, fugw, d)

  Calculate reduced tangent plane distance.
- subroutine, public set_stability_tolerance (relativetolerance)

  Set tolerance for stability solver.
- real function, public get_stability_tolerance ()

  Get tolerance for stability solver.

Variables

- real, parameter, public **stabilitylimit** = -machine_prec ∗ 1000.0

  Values > negative_lim is treated as zero or positive.

5.56.1 Detailed Description

Minimize reduced tangent plane distance.

Todo Add termination when approaching trivial solution.

5.56.2 Function/Subroutine Documentation

5.56.2.1 checkvlestability()

```
subroutine, public stability::checkvlestability (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             logical, intent(out) isstable,
             real, dimension(nc), intent(out) wsol,
             integer, intent(out) new_phase )
```

Check if a feed is stable against vapor-liquid split.

Author

Ailo, 2016-12-21

Parameters

in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*z*	Trial composition (Overall compozition)
out	*isstable*	Is the solution trivial? (That is; W == Z)
out	*wsol*	Solution.

5.56.2.2 get_stability_tolerance()

```
real function, public stability::get_stability_tolerance
```

Get tolerance for stability solver.

Author

MH, 2014-11

Returns

Relative tolerance for staility solver

5.56.2.3 set_stability_tolerance()

```
subroutine, public stability::set_stability_tolerance (
             real, intent(in) relativetolerance )
```

Set tolerance for stability solver.

Author

MH, 2014-11

Parameters

in	*relativetolerance*	Relative tolerance for staility solver

5.56.2.4 stabcalc()

```
real function, public stability::stabcalc (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             integer, intent(in) phase,
             real, dimension(nc), intent(in) fugz,
             real, dimension(nc), intent(out) fug,
             real, dimension(nc), intent(out), optional wsol,
             real, intent(in), optional pretermlim )
```

Calculate minimum reduced tangent plane distance.
Set starting compozition based on phase flag.
Note that the FUGZ must contain the single phase fugasities with the lowest gibbs energy when calling this function.

Author

MHA, 2012-01-30

Parameters

in	*phase*	Phase flag for compozition initiation (the
in	*z*	Trial composition (Overall compozition)
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*fugz*	Sigle phase fagasities for Z. Minimum gibbs solution.
out	*fug*	The fugasities at the solution, W.
out	*wsol*	Solution.
in	*pretermlim*	Control when solver terminates prematurely

Returns

Tangent plane distance

5.56.2.5 stabcalcw()

```
real function, public stability::stabcalcw (
             integer, intent(in) nd,
             integer, intent(in) k,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nd,nc), intent(in) xx,
             real, dimension(nc), intent(inout) w,
             integer, intent(in) phase,
             real, dimension(nc), intent(in) fugz,
             real, dimension(nc), intent(out) fugw,
             real, intent(in), optional pretermlim )
```

Calculate minimum reduced tangent plane distance.
$tpdm(W) = 1 + \sum W_i(\ln W_i + \ln \varphi - d_i - 1)$.
Where the sum is over all components.
Start by doing 3 iterations with successive substitution, if not converged, switch to a modefied newton minimizer.

Note that the FUGZ must contain the single phase fugasities with the lowest gibbs energy when calling this function. Initial compozition guess for W must also be supplied.

Author

MHA, 2012-01-30

Parameters

in	*nd*	Dimension of compozition matrix
in	*k*	Index of Trial phase
in	*phase*	Phase flag. Determine what root we are calculating in the eos.
in	*xx*	All phases in equilibrium
in,out	*w*	Inital guess
in	*fugz*	Fugacity of trial phase
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
out	*fugw*	The fugasities at the solution, W.
in	*pretermlim*	Control when solver terminates prematurely

5.56.2.6 tpd_fun()

```
real function, public stability::tpd_fun (
             real, dimension(nc), intent(in) w,
             real, dimension(nc), intent(in) fugw,
             real, dimension(nc), intent(in) d )
```

Calculate reduced tangent plane distance.
$tpdm(W) = 1 + \sum W_i(\ln W_i + \ln \varphi - d_i - 1)$.
Where the sum is over all components.

Author

MH, 2014-10-27

Parameters

in	*w*	Composition vector
in	*fugw*	The fugasities at the solution, W.

5.57 state_functions Module Reference

Calculate jacobian for Michelsen state matrices.

Functions/Subroutines

- subroutine, public getstatefuncmatrix (t, p, z, betav, betal, x, y, spec, vspec, m, rhs, phase, simplematrix)

  Calculate jacobian matrix for Michelsen state functions Implemented for two phase PH, PS, PT and UV flash Included TV and PV for convenience.
- subroutine, public getstatefunc (t, p, z, betav, betal, x, y, spec, vspec, rhs, phase)

  Calculate RHS function values for Michelsen state functions Values correnspponds to subroutine getStateFuncMatirx.
- real function, public dhdt_twophase (t, p, z, betav, betal, x, y, ph)

  Calculate two-phase heat capacity at constant pressure.

- real function, public dhdp_twophase (t, p, z, betav, betal, x, y, ph)

  Calculate two-phase enthalpy change at constant temperature.
- real function, public dvdt_twophase (t, p, z, betav, betal, x, y, ph)

  Calculate two-phase temperature differential of specific volume at constant pressure.
- real function, public dvdp_twophase (t, p, z, betav, betal, x, y, ph)

  Calculate two-phase pressure differential of specific volume at constant temperature.
- subroutine, public dnvdx (t, p, z, betav, betal, x, y, dbetadt, dbetadp, dndt, dndp)

  Calculate temperature/pressure differential of mole number (and phase fraction) at constant pressure.
- subroutine, public getsvderivativestwophase (t, p, z, betav, betal, x, y, iphase, dsdp_v, dsdt_v, dvdp_s, dvdt_s)

  Calculate single- or two-phase derivatives of entropy and/or molar volume.
- subroutine, public getuvderivativestwophase (t, p, z, betav, betal, x, y, iphase, dudp_v, dudt_v, dvdp_u, dvdt↩_u)

  Calculate single- or two-phase derivatives of internal energy and/or molar volume.
- real function, public getjoulethompsoncoeff (t, p, z, betav, betal, x, y, phase, ph)

  Calculate two-phase Joule-Thompson coefficient dT/dP at constant enthalpy.
- real function, public dpdt_twophase (t, p, z, betav, betal, x, y, phase, ph)

  Calculate temperaure differential of pressure at constant volume.

5.57.1 Detailed Description

Calculate jacobian for Michelsen state matrices.

Todo Need trace-component functionallity.

5.57.2 Function/Subroutine Documentation

5.57.2.1 dhdp_twophase()

```
real function, public state_functions::dhdp_twophase (
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) z,
            real, intent(in) betav,
            real, intent(in) betal,
            real, dimension(nc), intent(in) x,
            real, dimension(nc), intent(in) y,
            integer, dimension(2), intent(in), optional ph )
```

Calculate two-phase enthalpy change at constant temperature.

Author

MH, 2014

Parameters

in	*betav*	Vapour phase molar fraction [-]
in	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in	*x*	Liquid molar compozition [-]
in	*y*	Vapour molar compozition [-]
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*ph*	Phase integers

5.57.2.2 dhdt_twophase()

```
real function, public state_functions::dhdt_twophase (
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) z,
            real, intent(in) betav,
            real, intent(in) betal,
            real, dimension(nc), intent(in) x,
            real, dimension(nc), intent(in) y,
            integer, dimension(2), intent(in), optional ph )
```

Calculate two-phase heat capacity at constant pressure.

Author

MHA, 2012-03-20

Parameters

in	*betav*	Vapour phase molar fraction [-]
in	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in	*x*	Liquid molar compozition [-]
in	*y*	Vapour molar compozition [-]
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*ph*	Phase integers

5.57.2.3 dnvdx()

```
subroutine, public state_functions::dnvdx (
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) z,
            real, intent(in) betav,
            real, intent(in) betal,
            real, dimension(nc), intent(in) x,
            real, dimension(nc), intent(in) y,
            real, intent(out), optional dbetadt,
            real, intent(out), optional dbetadp,
            real, dimension(nc), intent(out), optional dndt,
            real, dimension(nc), intent(out), optional dndp )
```

Calculate temperature/pressure differential of mole number (and phase fraction) at constant pressure.

Author

MH, 2015-01

Parameters

in	*betav*	Vapour phase molar fraction [-]
in	*betal*	Vapour phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in	*x*	Liquid molar compozition [-]
in	*y*	Vapour molar compozition [-]
in	*t*	Temperature [K]

Parameters

in	*p*	Pressure [Pa]
out	*dndp*	Mole based

5.57.2.4 dpdt_twophase()

```
real function, public state_functions::dpdt_twophase (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             real, intent(in) betav,
             real, intent(in) betal,
             real, dimension(nc), intent(in) x,
             real, dimension(nc), intent(in) y,
             integer, intent(in) phase,
             integer, dimension(2), intent(in), optional ph )
```

Calculate temperaure differential of pressure at constant volume.

Author

MH, 2015-10

Parameters

in	*betav*	Vapour phase molar fraction [-]
in	*betal*	Vapour phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in	*x*	Liquid molar compozition [-]
in	*y*	Vapour molar compozition [-]
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*phase*	Phase identifyer
in	*ph*	Phase integers

5.57.2.5 dvdp_twophase()

```
real function, public state_functions::dvdp_twophase (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             real, intent(in) betav,
             real, intent(in) betal,
             real, dimension(nc), intent(in) x,
             real, dimension(nc), intent(in) y,
             integer, dimension(2), intent(in), optional ph )
```

Calculate two-phase pressure differential of specific volume at constant temperature.

Author

MH, 2012-07-06

Parameters

in	*betav*	Vapour phase molar fraction [-]

Parameters

in	*betal*	Vapour phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in	*x*	Liquid molar compozition [-]
in	*y*	Vapour molar compozition [-]
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*ph*	Phase integers

5.57.2.6 dvdt_twophase()

```
real function, public state_functions::dvdt_twophase (
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) z,
            real, intent(in) betav,
            real, intent(in) betal,
            real, dimension(nc), intent(in) x,
            real, dimension(nc), intent(in) y,
            integer, dimension(2), intent(in), optional ph )
```

Calculate two-phase temperature differential of specific volume at constant pressure.

Author

MH, 2012-07-06

Parameters

in	*betav*	Vapour phase molar fraction [-]
in	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in	*x*	Liquid molar compozition [-]
in	*y*	Vapour molar compozition [-]
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*ph*	Phase integers

5.57.2.7 getjoulethompsoncoeff()

```
real function, public state_functions::getjoulethompsoncoeff (
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) z,
            real, intent(in) betav,
            real, intent(in) betal,
            real, dimension(nc), intent(in) x,
            real, dimension(nc), intent(in) y,
            integer, intent(in) phase,
            integer, dimension(2), intent(in), optional ph )
```

Calculate two-phase Joule-Thompson coefficient dT/dP at constant enthalpy.

Author

MH, 2015-09

Parameters

in	*betav*	Vapour phase molar fraction [-]
in	*betal*	Vapour phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in	*x*	Liquid molar compozition [-]
in	*y*	Vapour molar compozition [-]
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*phase*	Phase identifyer
in	*ph*	Phase integers

5.57.2.8 getstatefunc()

```
subroutine, public state_functions::getstatefunc (
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) z,
            real, intent(in) betav,
            real, intent(in) betal,
            real, dimension(nc), intent(in) x,
            real, dimension(nc), intent(in) y,
            character(len=2), intent(in) spec,
            real, dimension(2), intent(in) vspec,
            real, dimension(:), intent(out) rhs,
            integer, dimension(2), intent(in), optional phase )
```

Calculate RHS function values for Michelsen state functions Values corrensponds to subroutine getStateFuncMatirx.

Author

MH, 2014-10-23

Parameters

in	*betav*	Vapour phase molar fraction [-]
in	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in	*x*	Liquid molar compozition [-]
in	*y*	Vapour molar compozition [-]
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*spec*	Character specification
in	*vspec*	Specification value
out	*rhs*	Right hand side
in	*phase*	Phase integer

5.57.2.9 getstatefuncmatrix()

```
subroutine, public state_functions::getstatefuncmatrix (
```

```
real, intent(in) t,
real, intent(in) p,
real, dimension(nc), intent(in) z,
real, intent(in) betav,
real, intent(in) betal,
real, dimension(nc), intent(in) x,
real, dimension(nc), intent(in) y,
character(len=2), intent(in) spec,
real, dimension(2), intent(in) vspec,
real, dimension(:,:), intent(out) m,
real, dimension(:), intent(out) rhs,
integer, dimension(2), intent(in), optional phase,
logical, intent(in), optional simplematrix )
```

Calculate jacobian matrix for Michelsen state functions Implemented for two phase PH, PS, PT and UV flash Included TV and PV for convenience.

Author

MHA, 2012-03-20

Parameters

in	*betav*	Vapour phase molar fraction [-]
in	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in	*x*	Liquid molar compozition [-]
in	*y*	Vapour molar compozition [-]
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*spec*	Character specification
in	*vspec*	Specification value
out	*m*	Jacobian matrix
out	*rhs*	Right hand side
in	*phase*	Phase integer
in	*simplematrix*	Deactivate composition diff.

5.57.2.10 getsvderivativestwophase()

```
subroutine, public state_functions::getsvderivativestwophase (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             real, intent(in) betav,
             real, intent(in) betal,
             real, dimension(nc), intent(in) x,
             real, dimension(nc), intent(in) y,
             integer, intent(in) iphase,
             real, intent(out), optional dsdp_v,
             real, intent(out), optional dsdt_v,
             real, intent(out), optional dvdp_s,
             real, intent(out), optional dvdt_s )
```

Calculate single- or two-phase derivatives of entropy and/or molar volume.

Author

MAG, 2015-03-03

Parameters

in	*betav*	Vapour phase molar fraction [-]
in	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in	*x*	Liquid molar compozition [-]
in	*y*	Vapour molar compozition [-]
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*iphase*	Phase flag
out	*dsdp_v*	Partial derivative of entropy w.r.t. pressure at constant molar volume
out	*dsdt_v*	Partial derivative of entropy w.r.t. temperature at constant molar volume
out	*dvdp_s*	Partial derivative of molar volume w.r.t. pressure at constant entropy
out	*dvdt_s*	Partial derivative of molar volume w.r.t. temperature at constant entropy

5.57.2.11 getuvderivativestwophase()

```
subroutine, public state_functions::getuvderivativestwophase (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             real, intent(in) betav,
             real, intent(in) betal,
             real, dimension(nc), intent(in) x,
             real, dimension(nc), intent(in) y,
             integer, intent(in) iphase,
             real, intent(out), optional dudp_v,
             real, intent(out), optional dudt_v,
             real, intent(out), optional dvdp_u,
             real, intent(out), optional dvdt_u )
```

Calculate single- or two-phase derivatives of internal energy and/or molar volume.
CAUTION: The implementations of dudp_v and dudt_v are covered by the unit tests in 3DMF. The implementations of dvdp_u and dvdt_u have not been tested yet.

Author

MAG, 2015-11-09

Parameters

in	*betav*	Vapour phase molar fraction [-]
in	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar composition [-]
in	*x*	Liquid molar composition [-]
in	*y*	Vapour molar composition [-]
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]

Parameters

in	*iphase*	Phase flag
out	*dudp_v*	Partial derivative of internal energy w.r.t. pressure at constant molar volume
out	*dudt_v*	Partial derivative of internal energy w.r.t. temperature at constant molar volume
out	*dvdp_u*	Partial derivative of molar volume w.r.t. pressure at constant internal energy
out	*dvdt_u*	Partial derivative of molar volume w.r.t. temperature at constant internal energy

5.58 sv_solver Module Reference

Calculate solve SV-flash for single phase gas/liquid or a gas-liquid mixture.

Functions/Subroutines

- subroutine, public twophasesvflash (t, p, z, beta, betal, x, y, sspec, vspec, phase, ierr)

  Do SV-flash: Switch for multicomponent (mc) and single component solver.
- subroutine, public twophasesvflashnested (t, p, z, beta, betal, x, y, sspec, vspec, phase, isconverged)

  Do SV-flash using PT-flash in nested loop.
- subroutine, public twophasesvflashfull (t, p, z, beta, betal, x, y, sspec, vspec, phase, converged)

  Do SV-flash using full equation system.
- subroutine, public fun_1ph_sv (f, var, param)

  Calculate state function for SV system.
- subroutine, public jac_1ph_sv (jac, var, param)

  Calculate state function for PH system and its differentials.
- subroutine, public enablecustumstabcalc (w, phase)

  Enable additional phase stability check.
- subroutine, public disablecustumstabcalc ()

  Disable additional phase stability check.
- subroutine, public twophasesvsinglecomp (t, p, z, beta, betal, x, y, sspec, vspec, phase, ierr)

  Do SV-flash for single component.
- subroutine, public singlecompsv_tv (t, p, z, beta, betal, x, y, sspec, vspec, phase, isconverged, pmin, pmax)

  Do SV-flash for single component single phase. Use U(T,v)
- subroutine, public setnestedsvtolerance (tolerance, nmax, linesearch_nmax)

  Set tolerance for nested loop SV flash.
- subroutine, public getnestedsvtolerance (tolerance, nmax, linesearch_nmax)

  Get tolerance for nested loop SV flash.
- subroutine, public setfulleqsvtolerance (tolerance, nmax, linesearch_nmax, gibbs_tolerance)

  Set tolerance for fulleq. SV flash.
- subroutine, public getfulleqsvtolerance (tolerance, nmax, linesearch_nmax, gibbs_tolerance)

  Get tolerance for fulleq. SV flash.
- subroutine, public setsinglecompsvtolerance (tolerance, nmax, linesearch_nmax)

  Set tolerance for single component SV flash.
- subroutine, public getsinglecompsvtolerance (tolerance, nmax, linesearch_nmax)

  Get tolerance for single component SV flash.

5.58.1 Detailed Description

Calculate solve SV-flash for single phase gas/liquid or a gas-liquid mixture.

Todo Need trace-component functionallity.

Consider merging with UV-flash

5.58.2 Function/Subroutine Documentation

5.58.2.1 disablecustumstabcalc()

```
subroutine, public sv_solver::disablecustumstabcalc
```

Disable additional phase stability check.

Author

MH, 2014-01

5.58.2.2 enablecustumstabcalc()

```
subroutine, public sv_solver::enablecustumstabcalc (
             real, dimension(nc), intent(in) w,
             integer, intent(in) phase )
```

Enable additional phase stability check.

Author

MH, 2014-01

Parameters

in	*phase*	Phase identifyer
in	*w*	Initial comosition in stability calculation

5.58.2.3 fun_1ph_sv()

```
subroutine, public sv_solver::fun_1ph_sv (
             real, dimension(2), intent(out) f,
             real, dimension(2), intent(in) var,
             real, dimension(nc+3), intent(in) param )
```

Calculate state function for SV system.

Author

MH, 2012-08-15

Parameters

out	*f*	Function values
in	*var*	Variable vector
in	*param*	Parameter vector

5.58.2.4 getfulleqsvtolerance()

```
subroutine, public sv_solver::getfulleqsvtolerance (
             real, intent(out) tolerance,
             integer, intent(out) nmax,
```

```
            integer, intent(out) linesearch_nmax,
            real, intent(out), optional gibbs_tolerance )
```

Get tolerance for fulleq. SV flash.

Author

MH, 2015

Parameters

out	*tolerance*	Solver tolerance
out	*nmax*	Maximum number of iteretions
out	*linesearch_nmax*	Maximum number of line-searches
out	*gibbs_tolerance*	Tolerance for when to accept two-phase solutions

5.58.2.5 getnestedsvtolerance()

```
subroutine, public sv_solver::getnestedsvtolerance (
            real, intent(out) tolerance,
            integer, intent(out) nmax,
            integer, intent(out) linesearch_nmax )
```

Get tolerance for nested loop SV flash.

Author

MH, 2015

Parameters

out	*tolerance*	Solver tolerance
out	*nmax*	Maximum number of iteretions
out	*linesearch_nmax*	Maximum number of line-searches

5.58.2.6 getsinglecompsvtolerance()

```
subroutine, public sv_solver::getsinglecompsvtolerance (
            real, intent(out) tolerance,
            integer, intent(out) nmax,
            integer, intent(out) linesearch_nmax )
```

Get tolerance for single component SV flash.

Author

MH, 2015

Parameters

out	*tolerance*	Solver tolerance
out	*nmax*	Maximum number of iteretions
out	*linesearch_nmax*	Maximum number of line-searches

5.58.2.7 jac_1ph_sv()

```
subroutine, public sv_solver::jac_1ph_sv (
```

```
            real, dimension(2,2), intent(out) jac,
            real, dimension(2), intent(in) var,
            real, dimension(nc+3), intent(in) param )
```

Calculate state function for PH system and its differentials.

Author

MH, 2012-08-15

Parameters

in	*var*	Variable vector
in	*param*	Parameter vector
out	*jac*	Jacobian objective function

5.58.2.8 setfulleqsvtolerance()

```
subroutine, public sv_solver::setfulleqsvtolerance (
            real, intent(in) tolerance,
            integer, intent(in) nmax,
            integer, intent(in) linesearch_nmax,
            real, intent(in), optional gibbs_tolerance )
```

Set tolerance for fulleq. SV flash.

Author

MH, 2015

Parameters

in	*tolerance*	Solver tolerance
in	*nmax*	Maximum number of iteretions
in	*linesearch_nmax*	Maximum number of line-searches
in	*gibbs_tolerance*	Tolerance for when to accept two-phase solutions

5.58.2.9 setnestedsvtolerance()

```
subroutine, public sv_solver::setnestedsvtolerance (
            real, intent(in) tolerance,
            integer, intent(in) nmax,
            integer, intent(in) linesearch_nmax )
```

Set tolerance for nested loop SV flash.

Author

MH, 2015

Parameters

in	*tolerance*	Solver tolerance
in	*nmax*	Maximum number of iteretions
in	*linesearch_nmax*	Maximum number of line-searches

5.58.2.10 setsinglecompsvtolerance()

```
subroutine, public sv_solver::setsinglecompsvtolerance (
             real, intent(in) tolerance,
             integer, intent(in) nmax,
             integer, intent(in) linesearch_nmax )
```

Set tolerance for single component SV flash.

Author

MH, 2015

Parameters

in	*tolerance*	Solver tolerance
in	*nmax*	Maximum number of iteretions
in	*linesearch_nmax*	Maximum number of line-searches

5.58.2.11 singlecompsv_tv()

```
subroutine, public sv_solver::singlecompsv_tv (
             real, intent(inout) t,
             real, intent(inout) p,
             real, dimension(nc), intent(in) z,
             real, intent(inout) beta,
             real, intent(inout) betal,
             real, dimension(nc), intent(inout) x,
             real, dimension(nc), intent(inout) y,
             real, intent(in) sspec,
             real, intent(in) vspec,
             integer, intent(inout) phase,
             logical, intent(out), optional isconverged,
             real, intent(in), optional pmin,
             real, intent(in), optional pmax )
```

Do SV-flash for single component single phase. Use U(T,v)

Author

MH, 2014

Parameters

in,out	*beta*	Vapour phase molar fraction [-]
in,out	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in,out	*x*	Liquid molar compozition [-]
in,out	*y*	Vapour molar compozition [-]
in,out	*t*	Temperature [K]
in,out	*p*	Pressure [Pa]
in	*sspec*	Specified entropy [J/mol/K]
in	*vspec*	Specified specific volume [m3/mol]
in,out	*phase*	Phase identifier

5.58.2.12 twophasesvflash()

```
subroutine, public sv_solver::twophasesvflash (
             real, intent(inout) t,
             real, intent(inout) p,
             real, dimension(nc), intent(in) z,
             real, intent(inout) beta,
             real, intent(out) betal,
             real, dimension(nc), intent(inout) x,
             real, dimension(nc), intent(inout) y,
             real, intent(in) sspec,
             real, intent(in) vspec,
             integer, intent(inout) phase,
             integer, intent(out), optional ierr )
```

Do SV-flash: Switch for multicomponent (mc) and single component solver.

Author

MH, 2014

Parameters

in,out	*beta*	Vapour phase molar fraction [-]
out	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in,out	*x*	Liquid molar compozition [-]
in,out	*y*	Vapour molar compozition [-]
in,out	*t*	Temperature [K]
in,out	*p*	Pressure [Pa]
in	*sspec*	Specified entropy [J/mol/K]
in	*vspec*	Specified specific volume [m3/mol]
in,out	*phase*	Phase identifier
out	*ierr*	Integer error flag

5.58.2.13 twophasesvflashfull()

```
subroutine, public sv_solver::twophasesvflashfull (
             real, intent(inout) t,
             real, intent(inout) p,
             real, dimension(nc), intent(in) z,
             real, intent(inout) beta,
             real, intent(out) betal,
             real, dimension(nc), intent(inout) x,
             real, dimension(nc), intent(inout) y,
             real, intent(in) sspec,
             real, intent(in) vspec,
             integer, intent(inout) phase,
             logical, intent(out) converged )
```

Do SV-flash using full equation system.
Assume initial values for specified phase.

Author

MH, 2012-08-15

Parameters

in,out	*beta*	Vapour phase molar fraction [-]
out	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in,out	*x*	Liquid molar compozition [-]
in,out	*y*	Vapour molar compozition [-]
in,out	*t*	Temperature [K]
in,out	*p*	Pressure [Pa]
in	*sspec*	Specified entropy [J/mol/K]
in	*vspec*	Specified specific volume [m3/mol]
in,out	*phase*	Phase identifier

5.58.2.14 twophasesvflashnested()

```
subroutine, public sv_solver::twophasesvflashnested (
             real, intent(inout) t,
             real, intent(inout) p,
             real, dimension(nc), intent(in) z,
             real, intent(inout) beta,
             real, intent(out) betal,
             real, dimension(nc), intent(inout) x,
             real, dimension(nc), intent(inout) y,
             real, intent(in) sspec,
             real, intent(in) vspec,
             integer, intent(inout) phase,
             logical, intent(out), optional isconverged )
```

Do SV-flash using PT-flash in nested loop.
Function to minimize: $-\frac{g_{min}+TS_{spec}-pv_{spec}}{R}$.

Author

MH, 2015-02

Parameters

in,out	*beta*	Vapour phase molar fraction [-]
out	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in,out	*x*	Liquid molar compozition [-]
in,out	*y*	Vapour molar compozition [-]
in,out	*t*	Temperature [K]
in,out	*p*	Pressure [Pa]
in	*sspec*	Specified entropy [J/mol/K]
in	*vspec*	Specified specific volume [m3/mol]
in,out	*phase*	Phase identifier

5.58.2.15 twophasesvsinglecomp()

```
subroutine, public sv_solver::twophasesvsinglecomp (
             real, intent(inout) t,
             real, intent(inout) p,
```

```
real, dimension(nc), intent(in) z,
real, intent(inout) beta,
real, intent(inout) betal,
real, dimension(nc), intent(inout) x,
real, dimension(nc), intent(inout) y,
real, intent(in) sspec,
real, intent(in) vspec,
integer, intent(inout) phase,
integer, intent(out), optional ierr )
```

Do SV-flash for single component.

Todo Need handling of solutions close to critical point

Author

MH, 2014

Parameters

in,out	*beta*	Vapour phase molar fraction [-]
in,out	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in,out	*x*	Liquid molar compozition [-]
in,out	*y*	Vapour molar compozition [-]
in,out	*t*	Temperature [K]
in,out	*p*	Pressure [Pa]
in	*sspec*	Specified entropy [J/mol/K]
in	*vspec*	Specified specific volume [m3/mol]
in,out	*phase*	Phase identifier
out	*ierr*	Phase identifier

5.59 thermo_utils Module Reference

Module for thermodynamic helper-routines which may be useful for user-applications, but which do not belong in eos.f90.

Functions/Subroutines

- integer function, public guessphase (t, p, z, t_comp, p_comp, vb_ratio)

  Estimate the single phase given by T,P,z as either liquid or vapor.
- integer function, public guessphasetv (t, v, z, t_comp, v_comp, vb_ratio)

  Estimate the single phase given by T,V,z as either liquid or vapor.
- logical function, public **iswatercomponent** (i)
- real function, public **watercomponentfraction** (z)
- subroutine, public calclnphioffset (pid, lnphi_offset)

  Set offset to Wilson liquid fugacity.
- subroutine, public wilsonk (t, p, k, dkdt, dkdp, liqtype)

  Calculate Wilson K-values.
- subroutine, public wilsonki (i, t, p, lnphi_offset, k)

  Calculate Wilson K-value for component i.
- subroutine, public wilsonkdiff (t, p, k, dkdp, dkdt)

  Calculate Wilson K-values with pressure and temperature differentials.

- integer function, public get_n_solids (nd, phasevec)

  Get number of solids in a mixture.
- logical function, public issinglecomp (z)

  Is the "mixture" single component.
- logical function, public istwocomp (z)

  Is the "mixture" two component.
- integer function, public maxcomp (z)

  Return index of largest component fraction.
- logical function, public phase_is_fake (t, p, z, phase)

  Is the solver returning FAKEPH solution.

5.59.1 Detailed Description

Module for thermodynamic helper-routines which may be useful for user-applications, but which do not belong in eos.f90.

Author

EA, 2014-05

5.59.2 Function/Subroutine Documentation

5.59.2.1 calclnphioffset()

```
subroutine, public thermo_utils::calclnphioffset (
             integer, intent(in), optional pid,
             real, dimension(nc), intent(out) lnphi_offset )
```

Set offset to Wilson liquid fugacity.

Author

MH, 2016-03

5.59.2.2 get_n_solids()

```
integer function, public thermo_utils::get_n_solids (
             integer nd,
             integer, dimension(nph), intent(in) phasevec )
```

Get number of solids in a mixture.

Author

MH, 2018-06

Parameters

	nd	Number of solids
in	*phasevec*	Phases

5.59.2.3 guessphase()

```
integer function, public thermo_utils::guessphase (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             real, intent(in), optional t_comp,
             real, intent(in), optional p_comp,
             real, intent(in), optional vb_ratio )
```

Estimate the single phase given by T,P,z as either liquid or vapor.

Author

EA, 2014-05. Updated EA, 2015-01 Based on original code by MH, 2012-11-22

Parameters

in	*t*	Temperature (K)
in	*p*	Pressure (Pa)
in	*z*	Molar composition (mol/mol)
in	*t_comp*	Override for comparison point (K)
in	*p_comp*	Override for comparison point (Pa)
in	*vb_ratio*	Override volume-covolume ratio

Returns

Best guess for phase (LIQPH or VAPPH)

5.59.2.4 guessphasetv()

```
integer function, public thermo_utils::guessphasetv (
             real, intent(in) t,
             real, intent(in) v,
             real, dimension(nc), intent(in) z,
             real, intent(in), optional t_comp,
             real, intent(in), optional v_comp,
             real, intent(in), optional vb_ratio )
```

Estimate the single phase given by T,V,z as either liquid or vapor.

Author

MH, 20018-04-13 Based on guessPhase code

Parameters

in	*t*	Temperature (K)
in	*v*	Volume (m3/mol)
in	*z*	Molar composition (mol/mol)
in	*t_comp*	Override for comparison point (K)
in	*v_comp*	Override for comparison point (m3/mol)
in	*vb_ratio*	Override volume-covolume ratio

Returns

Best guess for phase (LIQPH or VAPPH)

5.59.2.5 issinglecomp()

```
logical function, public thermo_utils::issinglecomp (
             real, dimension(nc), intent(in) z )
```

Is the "mixture" single component.

Author

M. Hammer October 2014

5.59.2.6 istwocomp()

```
logical function, public thermo_utils::istwocomp (
            real, dimension(nc), intent(in) z )
```

Is the "mixture" two component.

Author

M. Hammer March 2016

5.59.2.7 maxcomp()

```
integer function, public thermo_utils::maxcomp (
            real, dimension(nc), intent(in) z )
```

Return index of largest component fraction.

Author

M. Hammer October 2014

5.59.2.8 phase_is_fake()

```
logical function, public thermo_utils::phase_is_fake (
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) z,
            integer, intent(in) phase )
```

Is the solver returning FAKEPH solution.

Author

MH, 2020-01

Parameters

in	*z*	Molar compozition [-]
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]

Returns

Phase is FAKEPH

5.59.2.9 wilsonk()

```
subroutine, public thermo_utils::wilsonk (
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(out) k,
            real, dimension(nc), intent(out), optional dkdt,
            real, dimension(nc), intent(out), optional dkdp,
            integer, intent(in), optional liqtype )
```

Calculate Wilson K-values.

Author

MH, 2013-03-06

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure
out	*k*	K-values
out	*dkdt*	1/K - Differential of K-values wrpt. temperature
out	*dkdp*	1/Pa - Differential of K-values wrpt. pressure
in	*liqtype*	Type of liquid (WATER, NONWATER)

5.59.2.10 wilsonkdiff()

```
subroutine, public thermo_utils::wilsonkdiff (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(out) k,
             real, dimension(nc), intent(out) dkdp,
             real, dimension(nc), intent(out) dkdt )
```

Calculate Wilson K-values with pressure and temperature differentials.

Author

MH, 2013-03-06

Parameters

in	*t*	K - Temperature
in	*p*	Pa - Pressure
out	*k*	K-values
out	*dkdt*	1/K - Differential of K-values wrpt. temperature
out	*dkdp*	1/Pa - Differential of K-values wrpt. pressure

5.59.2.11 wilsonki()

```
subroutine, public thermo_utils::wilsonki (
             integer, intent(in) i,
             real, intent(in) t,
             real, intent(in) p,
             real, intent(in) lnphi_offset,
             real, intent(out) k )
```

Calculate Wilson K-value for component i.

Author

MH, 2013-03-06

Parameters

in	*i*	Component
in	*t*	K - Temperature
in	*p*	Pa - Pressure
out	*k*	K-value

5.60 thermopack_var Module Reference

Global variables for ThermoPack. They are initialized in the thermo_model module.

Data Types

- interface allocate_and_init_intf
- interface assign_intf
- type base_eos_param
- type eos_param_pointer
- type thermo_model
- type thermo_model_pointer

Functions/Subroutines

- type(thermo_model) function, pointer, public **get_active_thermo_model** ()
- logical function, public **active_thermo_model_is_associated** ()
- class(base_eos_param) function, pointer, public **get_active_eos** ()
- class(base_eos_param) function, pointer, public **get_active_alt_eos** ()
- type(gendata_pointer) function, dimension(:), pointer **get_active_comps** ()
- logical function **is_model_container** (model, index)
- subroutine, public **activate_model** (index)
- integer function, public **add_eos** ()
- subroutine, public **delete_eos** (index)
- subroutine, public **base_eos_dealloc** (eos)
- subroutine **assign_base_eos_param** (this, other)
- subroutine **thermo_model_dealloc** (model)
- subroutine, public **delete_all_eos** ()
- subroutine, public **get_eos_identification** (eosid)
- subroutine, public **set_tmin** (tmin)
- real function, public **get_tmin** ()
- subroutine, public **set_tmax** (tmax)
- real function, public **get_tmax** ()
- subroutine, public **set_pmin** (pmin)
- real function, public **get_pmin** ()
- subroutine, public **set_pmax** (pmax)
- real function, public **get_pmax** ()
- real function, public **get_rgas** ()
- subroutine, public **apparent_to_real_mole_numbers** (n, ne)
- subroutine, public **real_to_apparent_diff** (fe_n, f_n)
- subroutine, public **real_to_apparent_differentials** (fe_n, fe_tn, fe_vn, fe_nn, f_n, f_tn, f_vn, f_nn)
- subroutine, public tp_lnfug_apparent (nc, ne, n, p, lnfug_real, lnfug, dlnfugdt_real, dlnfugdp_real, dlnfugdn↩ _real, dlnfugdt, dlnfugdp, dlnfugdn)

Variables

- real **rgas** = Rgas_default

  J/mol/K.
- real **krgas** = 1000.0∗Rgas_default

  J/kmol/K Temperature/pressure min/max values.
- real **tptmax** = 999.0

  K.
- real **tptmin** = 80.0

  K.
- real **tppmax** = 1.0e8

Pa.

- real **tppmin** = 1.0e1

  Pa.

- integer **nph** = 0

  Number of phases:

- integer nc = 0

  Number of apparent components:

- integer **nce** = 0

  Symmetrical upper left part of v_stoich.

- integer **ncsym** = 0

  Total number of associating sites.

- integer **numassocsites** = 0
- character(len=eosid_len), dimension(:), pointer **complist**

  List of component names.

- type(apparent_container), pointer **apparent** => NULL()
- type(thermo_model), pointer **p_active_model** => NULL()
- type(thermo_model_pointer), dimension(:), allocatable **thermo_models**

5.60.1 Detailed Description

Global variables for ThermoPack. They are initialized in the thermo_model module.

5.60.2 Function/Subroutine Documentation

5.60.2.1 tp_lnfug_apparent()

```
subroutine, public thermopack_var::tp_lnfug_apparent (
             integer, intent(in) nc,
             real, dimension(nce), intent(in) ne,
             real, dimension(nc), intent(in) n,
             real, intent(in) p,
             real, dimension(nce), intent(in) lnfug_real,
             real, dimension(nc), intent(out) lnfug,
             real, dimension(nce), intent(in), optional dlnfugdt_real,
             real, dimension(nce), intent(in), optional dlnfugdp_real,
             real, dimension(nce,nce), intent(in), optional dlnfugdn_real,
             real, dimension(nc), intent(out), optional dlnfugdt,
             real, dimension(nc), intent(out), optional dlnfugdp,
             real, dimension(nc,nc), intent(out), optional dlnfugdn )
```

Parameters

in	*n*	Apparent mole numbers [mols]
in	*p*	Pressure [Pa]
in	*ne*	Real mole numbers [mols]
in	*lnfug_real*	Log of real fugacities
out	*lnfug*	Log of apparent fugacity

5.60.3 Variable Documentation

5.60.3.1 nc

```
integer thermopack_var::nc = 0
```

Number of apparent components:
Number of real components, to support apparent composition mode. Always have: nce >= nc

5.61 tp_solver Module Reference

Solve TP flash problem. Look for single phase or a mixture of LV.

Functions/Subroutines

- subroutine, public twophasetpflash (t, p, z, beta, betal, phase, x, y)

  Given pressure and temperature calculate phase distribution. Start by doing successive substitutions with acceleration with the Dominant Eigenvalue Method every 5th iteration. If no solution is found in 10 iterations, a modified Newton is used to converge the two phase flash.
- subroutine, public **rr_successive_substitution_iteration** (t, p, z, k_in, sloppy, x, y, beta, k_out, fugl, fugv, g_simp, converged, rr_has_solution, betal, phasey)

  Do one iteration of successive substitution. First, solve the Rachford- Rice equation for the given Z and K. Then, calculate the logarithm of the fugacity coefficients and update K.
- logical function, public rr_solve (nc_rr, z, k, beta, x, y, sloppy, betal)

  Given K-values for all components solve for vapour fraction. If a solution exsist the function will return true, otherwise false.
- real function, public objective (v, param)

  Calculate Gibbs energy for a liquid-vapour mixture.
- subroutine, public differentials (v, param, of, dofdv, h)

  Calculate Gibbs energy for a liquid-vapour mixture. And its differentials.

Variables

- real, public **g_tolerance** = machine_prec $*$ 10.0

  Accept two-phase solution even though its Gibbs energy is slightly larger than that of the single-phase feed.

5.61.1 Detailed Description

Solve TP flash problem. Look for single phase or a mixture of LV.

Todo Need trace-component functionallity.

Add DEM of order 2 for accelration of multi-phase Rachford-Rice

5.61.2 Function/Subroutine Documentation

5.61.2.1 differentials()

```
subroutine, public tp_solver::differentials (
             real, dimension(nc), intent(in) v,
             real, dimension(2*nc+2), intent(in) param,
             real, intent(out) of,
             real, dimension(nc), intent(out) dofdv,
             real, dimension(nc,nc), intent(out) h )
```

Calculate Gibbs energy for a liquid-vapour mixture. And its differentials.

Author

MHA, 2012-01-30

Parameters

in	*v*	Vapour mole numbers [mole]
in	*param*	Parameter vector
out	*h*	Hessian matrix of objective function
out	*dofdv*	Differential of objective function with respect to variables (V)
out	*of*	Objective function value

5.61.2.2 objective()

```
real function, public tp_solver::objective (
            real, dimension(nc), intent(in) v,
            real, dimension(2*nc+2), intent(in) param )
```

Calculate Gibbs energy for a liquid-vapour mixture.

Author

MHA, 2012-01-30

Parameters

in	*v*	Vapour mole numbers [mole]
in	*param*	Parameter vector

Returns

Objective function value

5.61.2.3 rr_solve()

```
logical function, public tp_solver::rr_solve (
            integer, intent(in) nc_rr,
            real, dimension(nc_rr), intent(in) z,
            real, dimension(nc_rr), intent(in) k,
            real, intent(out) beta,
            real, dimension(nc_rr), intent(out) x,
            real, dimension(nc_rr), intent(out) y,
            logical, intent(in) sloppy,
            real, intent(out) betal )
```

Given K-values for all components solve for vapour fraction. If a solution exsist the function will return true, otherwise false.
The Rachford-Rice equation to be solved: $g(\beta) = \sum \frac{z_i(K_i-1)}{1-\beta+\beta K_i} = 0$.
Where the sum is over all components.
To avoid problems for $\beta \approx 1.0$, the problem is instead solved for $1.0 - \beta$.
g is a monotonous function in β, and is solved using a Newton method combined with bracketing.

Author

MHA, 2012-01-30

Parameters

in	*nc_rr*	Number of components
out	*beta*	Vapour phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in	*k*	Molar based K-values [-]
out	*x*	Liquid molar compozition [-]
out	*y*	Vapour molar compozition [-]
in	*sloppy*	Only do single iteration
out	*betal*	Liquid phase molar fraction [-]

Returns

True is a solution exsist

5.61.2.4 twophasetpflash()

```
subroutine, public tp_solver::twophasetpflash (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             real, intent(out) beta,
             real, intent(out) betal,
             integer, intent(out) phase,
             real, dimension(nc), intent(out) x,
             real, dimension(nc), intent(out) y )
```

Given pressure and temperature calculate phase distribution. Start by doing successive substitutions with acceleration with the Dominant Eigenvalue Method every 5th iteration. If no solution is found in 10 iterations, a modified Newton is used to converge the two phase flash.
In order to find the phase distribution with the lowest gibbs energy, the minimum single gibbs energy is comapred with the mixture gibbs energy.

Author

MHA, 2012-01-30, EA, 2013-07, MAG, 2013-09

Parameters

out	*beta*	Vapour phase molar fraction [-]
in	*z*	Overall molar compozition [-]
out	*x*	Liquid molar compozition [-]
out	*y*	Vapour molar compozition [-]
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
out	*phase*	Phase identefier
out	*betal*	Liquid phase molar fraction [-]

5.62 trend_solver Module Reference

Solve for trend density given pressure and temperature.

Functions/Subroutines

- subroutine, public trend_density (t, p, z, phase_in, rho, phase_found_out, metaextr)

  Solve for density given pressure, temperature and composition. Inteface RUB solver and local solver.
- logical function, public trend_phase_is_fake (t, p, z, phase)

  Is the solver returning FAKEPH solution.

Variables

- integer, parameter **trend_liq** =1
- integer, parameter **trend_vap** =2
- logical, public **userubdensitysolver** = .false.

5.62.1 Detailed Description

Solve for trend density given pressure and temperature.

5.62.2 Function/Subroutine Documentation

5.62.2.1 trend_density()

```
subroutine, public trend_solver::trend_density (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             integer, intent(in) phase_in,
             real, intent(out) rho,
             integer, intent(out), optional phase_found_out,
             logical, intent(in), optional metaextr )
```

Solve for density given pressure, temperature and composition. Inteface RUB solver and local solver.

Author

MH, 2015-10

Parameters

in	*z*	Overall molar compozition [-]
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*phase_in*	Desired phase
in	*metaextr*	Use extremum if no root
out	*rho*	Density [mol/m3]
out	*phase_found_out*	Phase actually found

5.62.2.2 trend_phase_is_fake()

```
logical function, public trend_solver::trend_phase_is_fake (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             integer, intent(in) phase )
```

Is the solver returning FAKEPH solution.

Author

MH, 2020-01

Parameters

in	*z*	Molar compozition [-]
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]

Returns

Phase is FAKEPH

5.63 uv_solver Module Reference

Calculate solve UV-flash for single phase gas/liquid or a gas-liquid mixture.

Functions/Subroutines

- subroutine, public twophaseuvflash (t, p, z, beta, betal, x, y, uspec, vspec, phase)

  Do UV-flash: Switch for multicomponent (mc) and single component solver.
- subroutine, public twophaseuvflashnested (t, p, z, beta, betal, x, y, uspec, vspec, phase, isconverged)

  Do UV-flash using PT-flash in nested loop.
- subroutine, public twophaseuvflashfull (t, p, z, beta, betal, x, y, uspec, vspec, phase, converged)

  Do UV-flash using full equation system.
- subroutine, public fun_1ph (f, var, param)

  Calculate state function for UV system.
- subroutine, public jac_1ph (jac, var, param)

  Calculate state function for PH system and its differentials.
- subroutine, public enablecustumstabcalc (w, phase)

  Enable additional phase stability check.
- subroutine, public disablecustumstabcalc ()

  Disable additional phase stability check.
- subroutine, public twophaseuvsinglecomp (t, p, z, beta, betal, x, y, uspec, vspec, phase, ierr)

  Do UV-flash for single component.
- subroutine, public setnesteduvtolerance (tolerance, nmax, linesearch_nmax)

  Set tolerance for nested loop UV flash.
- subroutine, public getnesteduvtolerance (tolerance, nmax, linesearch_nmax)

  Get tolerance for nested loop UV flash.
- subroutine, public setfullequvtolerance (tolerance, nmax, linesearch_nmax, gibbs_tolerance)

  Set tolerance for fulleq. UV flash.
- subroutine, public getfullequvtolerance (tolerance, nmax, linesearch_nmax, gibbs_tolerance)

  Get tolerance for fulleq. UV flash.
- subroutine, public setsinglecompuvtolerance (tolerance, nmax, linesearch_nmax)

  Set tolerance for single component UV flash.
- subroutine, public getsinglecompuvtolerance (tolerance, nmax, linesearch_nmax)

  Get tolerance for single component UV flash.

5.63.1 Detailed Description

Calculate solve UV-flash for single phase gas/liquid or a gas-liquid mixture.

Todo Need trace-component functionallity.

5.63.2 Function/Subroutine Documentation

5.63.2.1 disablecustumstabcalc()

```
subroutine, public uv_solver::disablecustumstabcalc
```

Disable additional phase stability check.

Author

MH, 2014-01

5.63.2.2 enablecustumstabcalc()

```
subroutine, public uv_solver::enablecustumstabcalc (
             real, dimension(nc), intent(in) w,
             integer, intent(in) phase )
```

Enable additional phase stability check.

Author

MH, 2014-01

Parameters

in	*phase*	Phase identifyer
in	*w*	Initial comosition in stability calculation

5.63.2.3 fun_1ph()

```
subroutine, public uv_solver::fun_1ph (
             real, dimension(2), intent(out) f,
             real, dimension(2), intent(in) var,
             real, dimension(nc+3), intent(in) param )
```

Calculate state function for UV system.

Author

MH, 2012-08-15

Parameters

out	*f*	Function values
in	*var*	Variable vector
in	*param*	Parameter vector

5.63.2.4 getfullequvtolerance()

```
subroutine, public uv_solver::getfullequvtolerance (
             real, intent(out) tolerance,
             integer, intent(out) nmax,
             integer, intent(out) linesearch_nmax,
             real, intent(out), optional gibbs_tolerance )
```

Get tolerance for fulleq. UV flash.

Author

MH, 2015

Parameters

out	*tolerance*	Solver tolerance
out	*nmax*	Maximum number of iteretions
out	*linesearch_nmax*	Maximum number of line-searches
out	*gibbs_tolerance*	Tolerance for when to accept two-phase solutions

5.63.2.5 getnesteduvtolerance()

```
subroutine, public uv_solver::getnesteduvtolerance (
             real, intent(out) tolerance,
             integer, intent(out) nmax,
             integer, intent(out) linesearch_nmax )
```

Get tolerance for nested loop UV flash.

Author

MH, 2015

Parameters

out	*tolerance*	Solver tolerance
out	*nmax*	Maximum number of iteretions
out	*linesearch_nmax*	Maximum number of line-searches

5.63.2.6 getsinglecompuvtolerance()

```
subroutine, public uv_solver::getsinglecompuvtolerance (
             real, intent(out) tolerance,
             integer, intent(out) nmax,
             integer, intent(out) linesearch_nmax )
```

Get tolerance for single component UV flash.

Author

MH, 2015

Parameters

out	*tolerance*	Solver tolerance
out	*nmax*	Maximum number of iteretions
out	*linesearch_nmax*	Maximum number of line-searches

5.63.2.7 jac_1ph()

```
subroutine, public uv_solver::jac_1ph (
             real, dimension(2,2), intent(out) jac,
             real, dimension(2), intent(in) var,
             real, dimension(nc+3), intent(in) param )
```

Calculate state function for PH system and its differentials.

Author

MH, 2012-08-15

Parameters

in	*var*	Variable vector
in	*param*	Parameter vector
out	*jac*	Jacobian objective function

5.63.2.8 setfullequvtolerance()

```
subroutine, public uv_solver::setfullequvtolerance (
             real, intent(in) tolerance,
             integer, intent(in) nmax,
             integer, intent(in) linesearch_nmax,
             real, intent(in), optional gibbs_tolerance )
```

Set tolerance for fulleq. UV flash.

Author

MH, 2015

Parameters

in	*tolerance*	Solver tolerance
in	*nmax*	Maximum number of iteretions
in	*linesearch_nmax*	Maximum number of line-searches
in	*gibbs_tolerance*	Tolerance for when to accept two-phase solutions

5.63.2.9 setnesteduvtolerance()

```
subroutine, public uv_solver::setnesteduvtolerance (
             real, intent(in) tolerance,
             integer, intent(in) nmax,
             integer, intent(in) linesearch_nmax )
```

Set tolerance for nested loop UV flash.

Author

MH, 2015

Parameters

in	*tolerance*	Solver tolerance
in	*nmax*	Maximum number of iteretions
in	*linesearch_nmax*	Maximum number of line-searches

5.63.2.10 setsinglecompuvtolerance()

```
subroutine, public uv_solver::setsinglecompuvtolerance (
             real, intent(in) tolerance,
             integer, intent(in) nmax,
             integer, intent(in) linesearch_nmax )
```

Set tolerance for single component UV flash.

Author

MH, 2015

Parameters

in	*tolerance*	Solver tolerance
in	*nmax*	Maximum number of iteretions
in	*linesearch_nmax*	Maximum number of line-searches

5.63.2.11 twophaseuvflash()

```
subroutine, public uv_solver::twophaseuvflash (
             real, intent(inout) t,
             real, intent(inout) p,
             real, dimension(nc), intent(in) z,
             real, intent(inout) beta,
             real, intent(out) betal,
             real, dimension(nc), intent(inout) x,
             real, dimension(nc), intent(inout) y,
             real, intent(in) uspec,
             real, intent(in) vspec,
             integer, intent(inout) phase )
```

Do UV-flash: Switch for multicomponent (mc) and single component solver.

Author

MH, 2014

Parameters

in,out	*beta*	Vapour phase molar fraction [-]
out	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in,out	*x*	Liquid molar compozition [-]
in,out	*y*	Vapour molar compozition [-]
in,out	*t*	Temperature [K]
in,out	*p*	Pressure [Pa]
in	*uspec*	Specified internal energy [J/mol]
in	*vspec*	Specified specific volume [m3/mol]
in,out	*phase*	Phase identifier

5.63.2.12 twophaseuvflashfull()

```
subroutine, public uv_solver::twophaseuvflashfull (
             real, intent(inout) t,
             real, intent(inout) p,
             real, dimension(nc), intent(in) z,
             real, intent(inout) beta,
             real, intent(out) betal,
             real, dimension(nc), intent(inout) x,
             real, dimension(nc), intent(inout) y,
             real, intent(in) uspec,
             real, intent(in) vspec,
             integer, intent(inout) phase,
             logical, intent(out) converged )
```

Do UV-flash using full equation system.
Assume initial values for specified phase.

Author

MH, 2012-08-15

Parameters

in,out	*beta*	Vapour phase molar fraction [-]

Parameters

out	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in,out	*x*	Liquid molar compozition [-]
in,out	*y*	Vapour molar compozition [-]
in,out	*t*	Temperature [K]
in,out	*p*	Pressure [Pa]
in	*uspec*	Specified internal energy [J/mol]
in	*vspec*	Specified specific volume [m3/mol]
in,out	*phase*	Phase identifier

5.63.2.13 twophaseuvflashnested()

```
subroutine, public uv_solver::twophaseuvflashnested (
             real, intent(inout) t,
             real, intent(inout) p,
             real, dimension(nc), intent(in) z,
             real, intent(inout) beta,
             real, intent(out) betal,
             real, dimension(nc), intent(inout) x,
             real, dimension(nc), intent(inout) y,
             real, intent(in) uspec,
             real, intent(in) vspec,
             integer, intent(inout) phase,
             logical, intent(out), optional isconverged )
```

Do UV-flash using PT-flash in nested loop.
Function to minimize: $-\frac{g_{min}-u_{spec}-pv_{spec}}{T}$.

Author

MH, 2012-07-05

Parameters

in,out	*beta*	Vapour phase molar fraction [-]
out	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in,out	*x*	Liquid molar compozition [-]
in,out	*y*	Vapour molar compozition [-]
in,out	*t*	Temperature [K]
in,out	*p*	Pressure [Pa]
in	*uspec*	Specified internal energy [J/mol]
in	*vspec*	Specified specific volume [m3/mol]
in,out	*phase*	Phase identifier

5.63.2.14 twophaseuvsinglecomp()

```
subroutine, public uv_solver::twophaseuvsinglecomp (
             real, intent(inout) t,
             real, intent(inout) p,
             real, dimension(nc), intent(in) z,
```

```
real, intent(inout) beta,
real, intent(inout) betal,
real, dimension(nc), intent(inout) x,
real, dimension(nc), intent(inout) y,
real, intent(in) uspec,
real, intent(in) vspec,
integer, intent(inout) phase,
integer, intent(out), optional ierr )
```

Do UV-flash for single component.

Todo Need handling of solutions close to critical point

Author

MH, 2014

Parameters

in,out	*beta*	Vapour phase molar fraction [-]
in,out	*betal*	Liquid phase molar fraction [-]
in	*z*	Overall molar compozition [-]
in,out	*x*	Liquid molar compozition [-]
in,out	*y*	Vapour molar compozition [-]
in,out	*t*	Temperature [K]
in,out	*p*	Pressure [Pa]
in	*uspec*	Specified internal energy [J/mol]
in	*vspec*	Specified specific volume [m3/mol]
in,out	*phase*	Phase identifier
out	*ierr*	Phase identifier

5.64 vls Module Reference

Interface handling both fluid and solid equation of state.

Data Types

- type state

  Thermo state, used for debugging.

Functions/Subroutines

- subroutine, public vlsthermo (t, p, z, phase, lnfug, lnfugt, lnfugp, lnfugx, ophase, metaextremum)

  Calculate fugasity coefficient and differentials given composition, temperature and pressure. Interface for vapour, liquid and solid.
- subroutine, public vlsspecificvolume (t, p, z, phase, v, dvdt, dvdp, dvdx)

  Calculate single-phase specific volume given composition, temperature and pressure for fluid and solid phases.
- subroutine, public vlsenthalpy (t, p, z, phase, h, dhdt, dhdp, dhdx)

  Calculate single-phase specific enthalpy given composition, temperature and pressure for fluid and solid phases.
- subroutine, public vlsentropy (t, p, z, phase, s, dsdt, dsdp, dsdx)

  Calculate single-phase specific entropy given composition, temperature and pressure for fluid and solid phases.
- real function, public mpentropy (nd, t, p, beta, xx, phase)

  Calculate multi-phase entropy given composition, temperature and pressure Unit: J/mol/K.

- real function, public mpenthalpy (nd, t, p, beta, xx, phase)

  Calculate multi-phase enthalpy given composition, temperature and pressure Unit: J/mol.
- real function, public mpspecificvolume (nd, t, p, beta, xx, phase)

  Calculate multi-phase entropy given composition, temperature and pressure Unit: J/mol/K.
- subroutine, public inversephasemappingvlws (z, betagas, y, gaspresent, betaliquid, x, liquidpresent, betawater, w, waterpresent, betasolid, ws, solidpresent, nd, beta, xx, phasevec)
- subroutine, public specificenthalpyvlws (nc, nph, t, p, z, betagas, y, gaspresent, betaliquid, x, liquidpresent, betawater, w, waterpresent, betasolid, ws, solidpresent, h)

  Get the specific enthalpy from VLWS variables. Wrapper for mpEnthalpy.
- subroutine, public specificvolumevlws (nc, nph, t, p, z, betagas, y, gaspresent, betaliquid, x, liquidpresent, betawater, w, waterpresent, betasolid, ws, solidpresent, v)

  Get the specific enthalpy from VLWS variables. Wrapper for mpSpecificVolume.
- subroutine, public specificentropyvlws (nc, nph, t, p, z, betagas, y, gaspresent, betaliquid, x, liquidpresent, betawater, w, waterpresent, betasolid, ws, solidpresent, s)

  Get the entropy from VLWS variables. Wrapper for mpEntropy.
- subroutine, public printcurrentphases (nd, t, p, z, xx, beta, phasevec)

  Print info about current phases Used for debugging.

5.64.1 Detailed Description

Interface handling both fluid and solid equation of state.

Author

MH, 2016-06.

5.64.2 Function/Subroutine Documentation

5.64.2.1 inversephasemappingvlws()

```
subroutine, public vls::inversephasemappingvlws (
             real, dimension(nc) z,
             real, intent(in) betagas,
             real, dimension(nc), intent(in) y,
             logical, intent(in) gaspresent,
             real, intent(in) betaliquid,
             real, dimension(nc), intent(in) x,
             logical, intent(in) liquidpresent,
             real, intent(in) betawater,
             real, dimension(nc), intent(in) w,
             logical, intent(in) waterpresent,
             real, intent(in) betasolid,
             real, dimension(nc), intent(in) ws,
             logical, intent(in) solidpresent,
             integer, intent(out) nd,
             real, dimension(nph), intent(out) beta,
             real, dimension(nph,nc), intent(out) xx,
             integer, dimension(nph), intent(out) phasevec )
```

Parameters

out	*nd*	Number of stabel phases found [-]
out	*beta*	Phase molar fractions [mol/mol]
out	*xx*	Phase molar compozition [mol/mol]
out	*phasevec*	Phase identifier. Not to be trused [-]

5.64.2.2 mpenthalpy()

```
real function, public vls::mpenthalpy (
             integer, intent(in) nd,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nph), intent(in) beta,
             real, dimension(nph,nc), intent(in) xx,
             integer, dimension(nph) phase )
```

Calculate multi-phase enthalpy given composition, temperature and pressure Unit: J/mol.

Author

MH, 2013-02

Parameters

in	*nd*	Numper of phases
in	*t*	K - Temperature
in	*p*	Pa - Pressure
	phase	Phase identifyer
in	*beta*	Phase fractions
in	*xx*	Composition

Returns

J/mol - Specifc enthalpy

5.64.2.3 mpentropy()

```
real function, public vls::mpentropy (
             integer, intent(in) nd,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nph), intent(in) beta,
             real, dimension(nph,nc), intent(in) xx,
             integer, dimension(nph) phase )
```

Calculate multi-phase entropy given composition, temperature and pressure Unit: J/mol/K.

Author

MH, 2013-02

Parameters

in	*nd*	Numper of phases
in	*t*	K - Temperature
in	*p*	Pa - Pressure
	phase	Phase identifyer
in	*beta*	Phase fractions
in	*xx*	Composition

Returns

J/mol/K - Specific entropy

5.64.2.4 mpspecificvolume()

```
real function, public vls::mpspecificvolume (
             integer, intent(in) nd,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nph), intent(in) beta,
             real, dimension(nph,nc), intent(in) xx,
             integer, dimension(nph) phase )
```

Calculate multi-phase entropy given composition, temperature and pressure Unit: J/mol/K.

Author

MH, 2013-02

Parameters

in	*nd*	Numper of phases
in	*t*	K - Temperature
in	*p*	Pa - Pressure
	phase	Phase identifyer
in	*beta*	Phase fractions
in	*xx*	Composition

Returns

m3/mol - Specifc volume

5.64.2.5 printcurrentphases()

```
subroutine, public vls::printcurrentphases (
             integer, intent(in) nd,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
             real, dimension(nph,nc), intent(in) xx,
             real, dimension(nph), intent(in) beta,
             integer, dimension(nph), intent(in) phasevec )
```

Print info about current phases Used for debugging.

Author

MH, 2018-04

Parameters

in	*nd*	Number of phases

5.64.2.6 specificenthalpyvlws()

```
subroutine, public vls::specificenthalpyvlws (
             integer, intent(in) nc,
             integer, intent(in) nph,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(nc), intent(in) z,
```

```
            real, intent(in) betagas,
            real, dimension(nc), intent(in) y,
            logical, intent(in) gaspresent,
            real, intent(in) betaliquid,
            real, dimension(nc), intent(in) x,
            logical, intent(in) liquidpresent,
            real, intent(in) betawater,
            real, dimension(nc), intent(in) w,
            logical, intent(in) waterpresent,
            real, intent(in) betasolid,
            real, dimension(nc), intent(in) ws,
            logical, intent(in) solidpresent,
            real, intent(out) h )
```

Get the specific enthalpy from VLWS variables. Wrapper for mpEnthalpy.

Author

HLS, 2018-08

Parameters

in	*nc*	Number of components
in	*nph*	Number of possible phases
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*z*	Overall molar compozition [-]
in	*betagas*	Gas phase molar fraction [-]
in	*y*	Gas phase molar composition [-]
in	*gaspresent*	Is gas phase detected?
in	*betaliquid*	Liquid phase molar fraction [-]
in	*x*	Liquid phase molar composition [-]
in	*liquidpresent*	Is liquid phase detected?
in	*betawater*	Water phase molar fraction [-]
in	*w*	Water phase molar composition [-]
in	*waterpresent*	Is water phase detected?
in	*betasolid*	Solid phase molar fraction [-]
in	*ws*	Solid phase molar composition [-]
in	*solidpresent*	Is solid phase detected?
out	*h*	Enthalpy [J/mol]

5.64.2.7 specificentropyvlws()

```
subroutine, public vls::specificentropyvlws (
            integer, intent(in) nc,
            integer, intent(in) nph,
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) z,
            real, intent(in) betagas,
            real, dimension(nc), intent(in) y,
            logical, intent(in) gaspresent,
            real, intent(in) betaliquid,
            real, dimension(nc), intent(in) x,
            logical, intent(in) liquidpresent,
```

```
            real, intent(in) betawater,
            real, dimension(nc), intent(in) w,
            logical, intent(in) waterpresent,
            real, intent(in) betasolid,
            real, dimension(nc), intent(in) ws,
            logical, intent(in) solidpresent,
            real, intent(out) s )
```

Get the entropy from VLWS variables. Wrapper for mpEntropy.

Author

HLS, 2018-08

Parameters

in	*nc*	Number of components
in	*nph*	Number of possible phases
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*z*	Overall molar compozition [-]
in	*betagas*	Gas phase molar fraction [-]
in	*y*	Gas phase molar composition [-]
in	*gaspresent*	Is gas phase detected?
in	*betaliquid*	Liquid phase molar fraction [-]
in	*x*	Liquid phase molar composition [-]
in	*liquidpresent*	Is liquid phase detected?
in	*betawater*	Water phase molar fraction [-]
in	*w*	Water phase molar composition [-]
in	*waterpresent*	Is water phase detected?
in	*betasolid*	Solid phase molar fraction [-]
in	*ws*	Solid phase molar composition [-]
in	*solidpresent*	Is solid phase detected?
out	*s*	Specific volume [J/(mol K)]

5.64.2.8 specificvolumevlws()

```
subroutine, public vls::specificvolumevlws (
            integer, intent(in) nc,
            integer, intent(in) nph,
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(nc), intent(in) z,
            real, intent(in) betagas,
            real, dimension(nc), intent(in) y,
            logical, intent(in) gaspresent,
            real, intent(in) betaliquid,
            real, dimension(nc), intent(in) x,
            logical, intent(in) liquidpresent,
            real, intent(in) betawater,
            real, dimension(nc), intent(in) w,
            logical, intent(in) waterpresent,
            real, intent(in) betasolid,
            real, dimension(nc), intent(in) ws,
            logical, intent(in) solidpresent,
```

```
            real, intent(out) v )
```

Get the specific enthalpy from VLWS variables. Wrapper for mpSpecificVolume.

Author

HLS, 2018-08

Parameters

in	*nc*	Number of components
in	*nph*	Number of possible phases
in	*t*	Temperature [K]
in	*p*	Pressure [Pa]
in	*z*	Overall molar compozition [-]
in	*betagas*	Gas phase molar fraction [-]
in	*y*	Gas phase molar composition [-]
in	*gaspresent*	Is gas phase detected?
in	*betaliquid*	Liquid phase molar fraction [-]
in	*x*	Liquid phase molar composition [-]
in	*liquidpresent*	Is liquid phase detected?
in	*betawater*	Water phase molar fraction [-]
in	*w*	Water phase molar composition [-]
in	*waterpresent*	Is water phase detected?
in	*betasolid*	Solid phase molar fraction [-]
in	*ws*	Solid phase molar composition [-]
in	*solidpresent*	Is solid phase detected?
out	*v*	Specific volume [m^3/mol]

5.64.2.9 vlsenthalpy()

```
subroutine, public vls::vlsenthalpy (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(1:nc), intent(in) z,
             integer, intent(in) phase,
             real, intent(out) h,
             real, intent(out), optional dhdt,
             real, intent(out), optional dhdp,
             real, dimension(1:nc), intent(out), optional dhdx )
```

Calculate single-phase specific enthalpy given composition, temperature and pressure for fluid and solid phases.

Author

MH, 2016-06

Parameters

in	*phase*	Phase identifyer
in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*z*	Compozition
out	*h*	J/mol - Specific enthalpy
out	*dhdt*	J/mol/K - Specific enthalpy differential wrpt. temperature

Parameters

out	*dhdp*	J/mol/Pa - Specific enthalpy differential wrpt. pressure
out	*dhdx*	J/mol - Specific enthalpy differential wrpt. mole numbers

5.64.2.10 vlsentropy()

```
subroutine, public vls::vlsentropy (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(1:nc), intent(in) z,
             integer, intent(in) phase,
             real, intent(out) s,
             real, intent(out), optional dsdt,
             real, intent(out), optional dsdp,
             real, dimension(1:nc), intent(out), optional dsdx )
```

Calculate single-phase specific entropy given composition, temperature and pressure for fluid and solid phases.

Author

MH, 2016-06

Parameters

in	*phase*	Phase identifyer
in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*z*	Compozition
out	*s*	J/mol/K - Specific entropy
out	*dsdt*	J/mol/K^2 - Specific entropy differential wrpt. temperature
out	*dsdp*	J/mol/K/Pa - Specific entropy differential wrpt. pressure
out	*dsdx*	J/mol/K - Specific enthalpy differential wrpt. mole numbers

5.64.2.11 vlsspecificvolume()

```
subroutine, public vls::vlsspecificvolume (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(1:nc), intent(in) z,
             integer, intent(in) phase,
             real, intent(out) v,
             real, intent(out), optional dvdt,
             real, intent(out), optional dvdp,
             real, dimension(1:nc), intent(out), optional dvdx )
```

Calculate single-phase specific volume given composition, temperature and pressure for fluid and solid phases.

Author

MH, 2016-06

Parameters

in	*phase*	Phase identifyer
in	*t*	K - Temperature

Parameters

in	*p*	Pa - Pressure
in	*z*	Compozition
out	*v*	m3/mol - Specific volume
out	*dvdt*	m3/mol/K - Specific volume differential wrpt. temperature
out	*dvdp*	m3/mol/Pa - Specific volume differential wrpt. pressure
out	*dvdx*	m3/mol - Specific volume differential wrpt. mole numbers

5.64.2.12 vlsthermo()

```
subroutine, public vls::vlsthermo (
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(1:nc), intent(in) z,
             integer, intent(in) phase,
             real, dimension(1:nc), intent(out) lnfug,
             real, dimension(1:nc), intent(out), optional lnfugt,
             real, dimension(1:nc), intent(out), optional lnfugp,
             real, dimension(1:nc,1:nc), intent(out), optional lnfugx,
             integer, intent(out), optional ophase,
             logical, intent(in), optional metaextremum )
```

Calculate fugasity coefficient and differentials given composition, temperature and pressure. Interface for vapour, liquid and solid.

Author

MH, 2016-06

Parameters

in	*phase*	Phase identifyer
out	*ophase*	Phase identifyer for MINGIBBSPH
in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*z*	Compozition
out	*lnfug*	Logarithm of fugasity coefficient
out	*lnfugt*	1/K - Logarithm of fugasity coefficient differential wrpt. temperature
out	*lnfugp*	1/Pa - Logarithm of fugasity coefficient differential wrpt. pressure
out	*lnfugx*	Logarithm of fugasity coefficient differential wrpt. mole numbers

5.65 volume_shift Module Reference

Calculate potenital corrections due to volume correction For documentation see memo: peneloux.pdf.

Functions/Subroutines

- integer function, public initvolumeshift (nc, comp, shiftid, eos, param_ref)

  Initialize volume shift parameters.
- real function, public eosvolumefromshiftedvolume (nc, comp, volumeshiftid, t, v, z)

  Get the volume to feed to the EoS, given the actual (shifted) volume.
- subroutine, public volumeshiftzfac (nc, comp, volumeshiftid, t, p, z, phase, zfac, dzdt, dzdp, dzdz)

Calculate volume shift.

- subroutine, public redefine_volume_shift (nc, j, comp, vlcurrent, vlexp)

  Redefine volume shift for component j.
- subroutine, public vshift_f_terms (nc, comp, volumeshiftid, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_vv, f_tn, f_vn, f_nn, f_vvv)

  Volume shift of residual, reduced Helmholtz energy, F.
- subroutine, public vshift_f_differential_dependencies (nc, volumeshiftid, include_f_v, include_f_tv, include_↩ f_vv, include_f_vn, f_t, f_v, f_n, f_tt, f_tv, f_vv, f_tn, f_vn, f_nn)

  Which additional differentials are needed for volume shift correction=.

Variables

- integer, parameter, public **noshift** =0

  Volume shift identifyers.
- integer, parameter, public **peneloux** =1

5.65.1 Detailed Description

Calculate potenital corrections due to volume correction For documentation see memo: peneloux.pdf.

Author

MH, June 2014

5.65.2 Function/Subroutine Documentation

5.65.2.1 eosvolumefromshiftedvolume()

```
real function, public volume_shift::eosvolumefromshiftedvolume (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc), intent(in) comp,
            integer, intent(in) volumeshiftid,
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(1:nc), intent(in) z )
```

Get the volume to feed to the EoS, given the actual (shifted) volume.

Author

MH, June 2014

Parameters

in	*volumeshiftid*	Volume shift identifier
in	*t*	K - Temperature
in	*v*	m3/mol - Actual, shifted volume
in	*z*	Composition

Returns

m3/mol - EoS volume

5.65.2.2 initvolumeshift()

```
integer function, public volume_shift::initvolumeshift (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc), intent(inout) comp,
```

```
            character(len=*), intent(in) shiftid,
            character(len=*), intent(in) eos,
            character(len=*), intent(in), optional param_ref )
```

Initialize volume shift parameters.

Author

MH, June 2014

Parameters

in	*nc*	Number of components
in,out	*comp*	Component data
in	*shiftid*	String volume shif identifyer
in	*eos*	Eos string
in	*param_ref*	Parameter set

5.65.2.3 redefine_volume_shift()

```
subroutine, public volume_shift::redefine_volume_shift (
            integer, intent(in) nc,
            integer, intent(in) j,
            type (gendata_pointer), dimension(nc), intent(inout) comp,
            real, intent(in) vlcurrent,
            real, intent(in) vlexp )
```

Redefine volume shift for component j.

Author

MH, May 2019

Parameters

in	*nc*	number of components
in	*j*	component
in,out	*comp*	Component data
in	*vlcurrent*	specific volume with current ci [m3/mol]
in	*vlexp*	experimental volume [m3/mol]

5.65.2.4 volumeshiftzfac()

```
subroutine, public volume_shift::volumeshiftzfac (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc), intent(in) comp,
            integer, intent(in) volumeshiftid,
            real, intent(in) t,
            real, intent(in) p,
            real, dimension(1:nc), intent(in) z,
            integer, intent(in) phase,
            real, intent(out) zfac,
            real, intent(out), optional dzdt,
            real, intent(out), optional dzdp,
            real, dimension(1:nc), intent(out), optional dzdz )
```

Calculate volume shift.

Author

MH, June 2014

Parameters

in	*volumeshiftid*	Volume shift identifyer
in	*phase*	Phase identifyer
in	*t*	K - Temperature
in	*p*	Pa - Pressure
in	*z*	Compozition
out	*zfac*	- Compressibillity factor
out	*dzdt*	1/K - Compressibillity factor differential wrpt. temperature
out	*dzdp*	1/Pa - Compressibillity factor differential wrpt. pressure
out	*dzdz*	1/mol - Compressibillity factor differential wrpt. mol numbers

5.65.2.5 vshift_f_differential_dependencies()

```
subroutine, public volume_shift::vshift_f_differential_dependencies (
            integer, intent(in) nc,
            integer, intent(in) volumeshiftid,
            logical, intent(out) include_f_v,
            logical, intent(out) include_f_tv,
            logical, intent(out) include_f_vv,
            logical, intent(out) include_f_vn,
            real, intent(inout), optional f_t,
            real, intent(inout), optional f_v,
            real, dimension(nc), intent(inout), optional f_n,
            real, intent(inout), optional f_tt,
            real, intent(inout), optional f_tv,
            real, intent(inout), optional f_vv,
            real, dimension(nc), intent(inout), optional f_tn,
            real, dimension(nc), intent(inout), optional f_vn,
            real, dimension(nc,nc), intent(inout), optional f_nn )
```

Which additional differentials are needed for volume shift correction=.

Author

Morten Hammer, April 2023

Parameters

in	*volumeshiftid*	Volume shift identifier

5.65.2.6 vshift_f_terms()

```
subroutine, public volume_shift::vshift_f_terms (
            integer, intent(in) nc,
            type (gendata_pointer), dimension(nc), intent(in) comp,
            integer, intent(in) volumeshiftid,
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(1:nc), intent(in) n,
            real, intent(inout), optional f,
            real, intent(inout), optional f_t,
```

```
real, intent(inout), optional f_v,
real, dimension(nc), intent(inout), optional f_n,
real, intent(inout), optional f_tt,
real, intent(inout), optional f_tv,
real, intent(inout), optional f_vv,
real, dimension(nc), intent(inout), optional f_tn,
real, dimension(nc), intent(inout), optional f_vn,
real, dimension(nc,nc), intent(inout), optional f_nn,
real, intent(inout), optional f_vvv )
```

Volume shift of residual, reduced Helmholtz energy, F.
Temperature-dependent volume shift not implemented, but easy to do

Author

Ailo, May 2020

Parameters

in	*volumeshiftid*	Volume shift identifier
in	*t*	K - Temperature
in	*v*	m3 - Volume
in	*n*	Mole numbers

5.66 wong_sandler Module Reference

Wong-Sandler equation of state Cubic EOS with temperature dependent b-parameter.

Functions/Subroutines

- real function, public **fidel_alpha** (tau12, tau21)

  Does calcualtions realted to the Wang-Sandler mixing model.
- subroutine, public **wongsandlermix** (cbeos, t, zcomp)

5.66.1 Detailed Description

Wong-Sandler equation of state Cubic EOS with temperature dependent b-parameter.
See: Wong, D. S. H. and Sandler, S. I. (1992). "A theoretically correct mixing rule for cubic equations of state". AIChE Journal 38: 671-680. doi: 10.1002/aic.690380505.
Equation reference in this module is to appendix equations A1 etc. of the Wong-Sandler paper.
See also separate memo regarding the model and differentials, located in the doc folder.

Author

AA, 2015-02, MH, 2015-03

Chapter 6

Data Type Documentation

6.1 hyperdual_mod::abs Interface Reference

Public Member Functions

- elemental type(hyperdual) function **abshyperdual** (v1)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.2 hyperdual_mod::acos Interface Reference

Public Member Functions

- elemental type(hyperdual) function **acoshyperdual** (v1)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.3 saturation_curve::aep Type Reference

Public Member Functions

- procedure, public **print** (a)

Public Attributes

- integer **type** = AZ_NONE
- logical **found** = .false.
- real **t** = 0
- real **p** = 0
- real **vg** = 0
- real **vl** = 0
- real, dimension(2) **x** = 0

The documentation for this type was generated from the following file:

- saturation_curve.f90

6.4 thermopack_var::allocate_and_init_intf Interface Reference

Public Member Functions

- subroutine allocate_and_init_intf (eos, nc, eos_label)

6.4.1 Constructor & Destructor Documentation

6.4.1.1 allocate_and_init_intf()

```
subroutine thermopack_var::allocate_and_init_intf::allocate_and_init_intf (
             class(base_eos_param), intent(inout) eos,
             integer, intent(in) nc,
             character(len=*), intent(in) eos_label )  [virtual]
```

Parameters

in	*nc*	Number of components
in	*eos_label*	EOS label

The documentation for this interface was generated from the following file:

- thermopack_var.f90

6.5 multiparameter_base::alpha0_hd_intf Interface Reference

Public Member Functions

- type(hyperdual) function alpha0_hd_intf (this, delta, tau)

6.5.1 Constructor & Destructor Documentation

6.5.1.1 alpha0_hd_intf()

```
type(hyperdual) function multiparameter_base::alpha0_hd_intf::alpha0_hd_intf (
             class(meos) this,
             type(hyperdual), intent(in) delta,
             type(hyperdual), intent(in) tau )  [virtual]
```

Parameters

in	*delta*	Reduced density (-)
in	*tau*	Reduced temperature (-)

Returns

Ideal reduced Helmholtz energy

The documentation for this interface was generated from the following file:

- multiparameter_base.f90

6.6 multiparameter_base::alpha0derivs_intf Interface Reference

Public Member Functions

- subroutine alpha0derivs_intf (this, delta, tau, alp0)

6.6.1 Constructor & Destructor Documentation

6.6.1.1 alpha0derivs_intf()

```
subroutine multiparameter_base::alpha0derivs_intf::alpha0derivs_intf (
             class(meos) this,
             real, intent(in) delta,
```

```
real, intent(in) tau,
real, dimension(0:2,0:2), intent(out) alp0 )  [virtual]
```

Parameters

in	*delta*	Reduced density (-)
in	*tau*	Reduced temperature (-)
out	*alp0*	alp0(i,j) = [(d_delta)^i(d_tau)^j alpha0]∗delta^i∗tau^j

The documentation for this interface was generated from the following file:

- multiparameter_base.f90

6.7 cubic_eos::alpha_label_mapping Type Reference

Public Attributes

- integer **alpha_idx**
- integer **n_param**
- character(len=short_label_len) **short_label**
- character(len=label_len) **description**
- integer **classic_for_eos_idx**

The documentation for this type was generated from the following file:

- cubic_eos.f90

6.8 compdata::alphadatadb Type Reference

Alpha correlation for cubic EoS.

Public Attributes

- character(len=uid_len) **cid**

  The component ID.
- character(len=ref_len) **ref**

  Data group reference.
- character(len=eosid_len) **eosid**

  EOS identifyer.
- real, dimension(3) **coeff**

6.8.1 Detailed Description

Alpha correlation for cubic EoS.
The documentation for this type was generated from the following file:

- compdata.f90

6.9 multiparameter_base::alphares_hd_intf Interface Reference

Public Member Functions

- type(hyperdual) function alphares_hd_intf (this, delta, tau)

6.9.1 Constructor & Destructor Documentation

6.9.1.1 alphares_hd_intf()

```
type(hyperdual) function multiparameter_base::alphares_hd_intf::alphares_hd_intf (
            class(meos) this,
            type(hyperdual), intent(in) delta,
            type(hyperdual), intent(in) tau )  [virtual]
```

Parameters

in	*delta*	Reduced density (-)
in	*tau*	Reduced temperature (-)

Returns

Residual reduced Helmholtz energy

The documentation for this interface was generated from the following file:

- multiparameter_base.f90

6.10 multiparameter_base::alpharesderivs_intf Interface Reference

Public Member Functions

- subroutine alpharesderivs_intf (this, delta, tau, alpr)

6.10.1 Constructor & Destructor Documentation

6.10.1.1 alpharesderivs_intf()

```
subroutine multiparameter_base::alpharesderivs_intf::alpharesderivs_intf (
            class(meos) this,
            real, intent(in) delta,
            real, intent(in) tau,
            real, dimension(0:2,0:2), intent(out) alpr )  [virtual]
```

Parameters

in	*delta*	Reduced density (-)
in	*tau*	Reduced temperature (-)
out	*alpr*	alpr(i,j) = [(d_delta)^i(d_tau)^j alphaRes]*delta^i*tau^j

The documentation for this interface was generated from the following file:

- multiparameter_base.f90

6.11 apparent_compostion::apparent_container Type Reference

Public Member Functions

- procedure, public **apparent_to_real_mole_numbers** (apparent, n, ne)

  Map apparent mole numbers to real mole numbers used by model.
- procedure, public **real_to_apparent_differentials** (apparent, fe_n, fe_tn, fe_vn, fe_nn, f_n, f_tn, f_vn, f_nn)

  Map from real mole number differentials to apparent mole number differentials.
- procedure, public **real_to_apparent_diff** (apparent, fe_n, f_n)

  Map from real mole number differential to apparent mole number differentials.
- procedure, public tp_lnfug_apparent (apparent, nc, ne, n, p, lnfug_real, lnfug, dlnfugdt_real, dlnfugdp_real, dlnfugdn_real, dlnfugdt, dlnfugdp, dlnfugdn)

  Convert logarithmic fugacity coefficient from real to apparent composition.
- procedure, public getmodfugacity (apparent, t, p, x, lnfug, sumne)

  Back calculate logaritm of fugacity coefficient, by removing dependency of log(x) and log(xe)
- procedure, public **update_v_sum** (apparent)
- procedure, public **set_v_stoich_ij** (apparent, i, j, v)
- procedure, public **dealloc** (apparent)

Public Attributes

- real, dimension(:,:), allocatable **v_stoich**
- real, dimension(:), allocatable **v_sum**
- integer **nc** =0
- integer **nce** =0
- integer **ncsym** =0

  Symmetrical upper left part of v_stoich.

6.11.1 Member Function/Subroutine Documentation

6.11.1.1 getmodfugacity()

```
procedure, public apparent_compostion::apparent_container::getmodfugacity (
             class(apparent_container), intent(in) apparent,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(apparent%nc), intent(in) x,
             real, dimension(apparent%nc), intent(inout) lnfug,
             real, intent(out) sumne )
```

Back calculate logaritm of fugacity coefficient, by removing dependency of log(x) and log(xe)

Author

MH, 2017-05

Parameters

in	*t*	Temperature (K)
in	*p*	Pressure (Pa)
in	*x*	Phase compozition

6.11.1.2 tp_lnfug_apparent()

```
procedure, public apparent_compostion::apparent_container::tp_lnfug_apparent (
             class(apparent_container), intent(in) apparent,
             integer, intent(in) nc,
             real, dimension(apparent%nce), intent(in) ne,
             real, dimension(nc), intent(in) n,
             real, intent(in) p,
             real, dimension(apparent%nce), intent(in) lnfug_real,
             real, dimension(nc), intent(out) lnfug,
             real, dimension(apparent%nce), intent(in), optional dlnfugdt_real,
             real, dimension(apparent%nce), intent(in), optional dlnfugdp_real,
             real, dimension(apparent%nce,apparent%nce), intent(in), optional dlnfugdn_real,
             real, dimension(nc), intent(out), optional dlnfugdt,
             real, dimension(nc), intent(out), optional dlnfugdp,
             real, dimension(nc,nc), intent(out), optional dlnfugdn )
```

Convert logarithmic fugacity coefficient from real to apparent composition.

Author

Morten Hammer, 2017-03

Parameters

in	*n*	Apparent mole numbers [mols]
in	*p*	Pressure [Pa]

Parameters

in	*ne*	Real mole numbers [mols]
in	*lnfug_real*	Log of real fugacities
out	*lnfug*	Log of apparent fugacity

The documentation for this type was generated from the following file:

- apparent_composition.f90

6.12 hyperdual_mod::asin Interface Reference

Public Member Functions

- elemental type(hyperdual) function **asinhyperdual** (v1)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.13 thermopack_var::assign_intf Interface Reference

Public Member Functions

- subroutine **assign_intf** (this, other)

The documentation for this interface was generated from the following file:

- thermopack_var.f90

6.14 multiparameter_base::assign_meos_intf Interface Reference

Public Member Functions

- subroutine **assign_meos_intf** (this, other)

The documentation for this interface was generated from the following file:

- multiparameter_base.f90

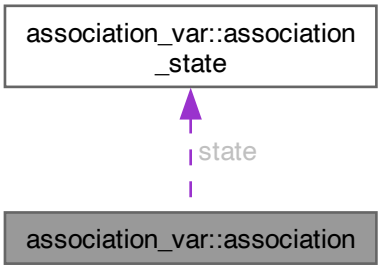
6.15 hyperdual_mod::assignment(=) Interface Reference

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.16 association_var::association Type Reference

Collaboration diagram for association_var::association:



Public Member Functions

- procedure, public **dealloc** (assoc)
- procedure, public **print** (assoc)

Public Attributes

- integer **saft_model**
- integer **active**
- integer **saft**
- integer **model**
- integer **set**
- integer **to**
- integer **the**
- integer **correct**
- integer **eos**
- integer **index**
- integer **stored**
- integer **in**
- integer **module**
- integer **eosdata**
- integer **numassocsites**

  Total number of associating sites.
- integer **numassoccomps**

  Number of associating components.
- integer, dimension(:), allocatable **compidcs**

  Component indices of associating components.
- integer, dimension(:,:), allocatable **comp_vs_sites**

  comp_vs_sites is an (nc)x2-matrix. Row i holds information on component number i. Column 1 and column 2 both equal the integer noSitesFlag if the component does not self-associate. If the component does self-associate, the rows hold the first and last association site number.
- real, dimension(:,:), allocatable beta_kl

  Model parameters that control the association strength.
- real, dimension(:,:), allocatable **eps_kl**

  Association energy.
- type(association_state) **state**

6.16.1 Member Data Documentation

6.16.1.1 beta_kl

`real, dimension(:,:), allocatable association_var::association::beta_kl`
Model parameters that control the association strength.
Effective association volume between site Ai and Bj (called \beta^{A_i B_j} in CPA).
The documentation for this type was generated from the following file:

- association_var.f90

6.17 association_var::association_state Type Reference

Current state for eos evaluation.

Public Member Functions

- procedure **init** (assoc_p, nc, t, v, n)
- procedure **init_fmt** (assoc_p, nc, t, n_fmt, m)
- procedure **dealloc** (assoc_p)

Public Attributes

- logical **fmt_mode** = .false.
- real **t**
- real **v**
- real, dimension(:), allocatable **n**
- real, dimension(:,:), allocatable **n_fmt**

6.17.1 Detailed Description

Current state for eos evaluation.
The documentation for this type was generated from the following file:

- association_var.f90

6.18 hyperdual_mod::atan Interface Reference

Public Member Functions

- elemental type(hyperdual) function **atanhyperdual** (v1)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.19 hyperdual_mod::atan2 Interface Reference

Public Member Functions

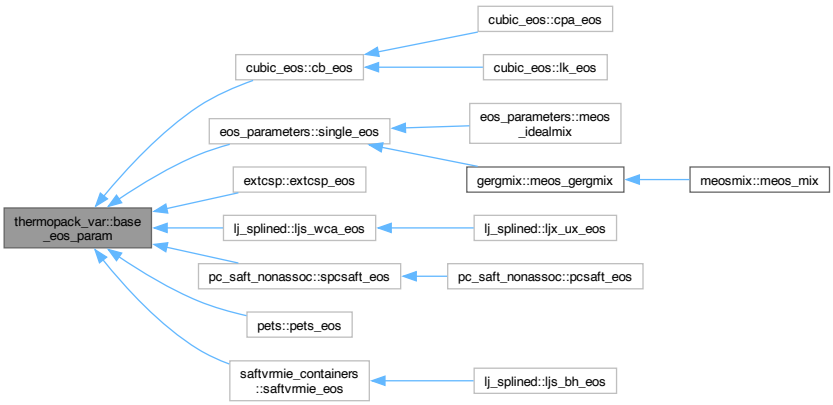
- elemental type(hyperdual) function **atan2hyperdual** (v1, v2)

The documentation for this interface was generated from the following file:

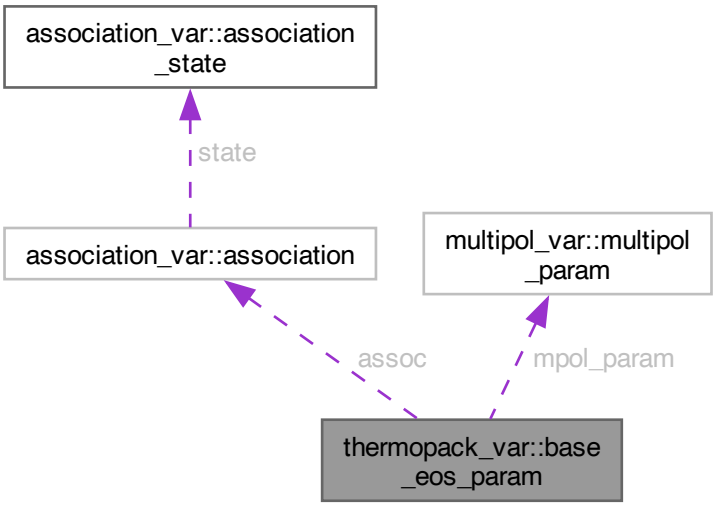
- hyperdual_mod.f90

6.20 thermopack_var::base_eos_param Type Reference

Inheritance diagram for thermopack_var::base_eos_param:



Collaboration diagram for thermopack_var::base_eos_param:



Public Member Functions

- procedure(allocate_and_init_intf), deferred, public allocate_and_init (eos, nc, eos_label)
- procedure, public **dealloc** (eos)
- procedure(assign_intf), deferred, pass, public **assign_eos** (this, other)
- generic, public **assignment** assign_eos
- procedure, public **assign_base_eos_param** (this, other)

Public Attributes

- character(len=eosid_len) **eosid**

Eos identifier.

- integer **eosidx**

  Eos group index.

- integer **subeosidx**

  Eos sub-index.

- integer **volumeshiftid** = 0

  0: No volume shift, 1:Peneloux shift

- logical **iselectrolyteeos** = .false.

  Used to enable electrolytes.

- type(association), pointer **assoc** => NULL()
- type(multipol_param), pointer **mpol_param** => NULL()

6.20.1 Member Function/Subroutine Documentation

6.20.1.1 allocate_and_init()

```
procedure(allocate_and_init_intf), deferred, public thermopack_var::base_eos_param::allocate↩
_and_init (
             class(base_eos_param), intent(inout) eos,
             integer, intent(in) nc,
             character(len=*), intent(in) eos_label )  [pure virtual]
```

Parameters

in	*nc*	Number of components
in	*eos_label*	EOS label

The documentation for this type was generated from the following file:

- thermopack_var.f90

6.21 c_interface_module::c_strlen Interface Reference

Interface to std C library function strlen.

Public Member Functions

- pure integer(c_size_t) function **c_strlen** (s)

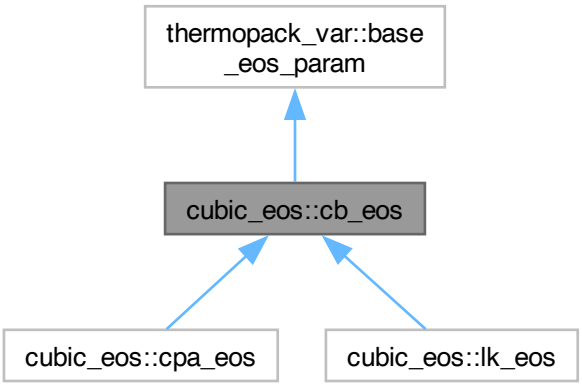
6.21.1 Detailed Description

Interface to std C library function strlen.
The documentation for this interface was generated from the following file:

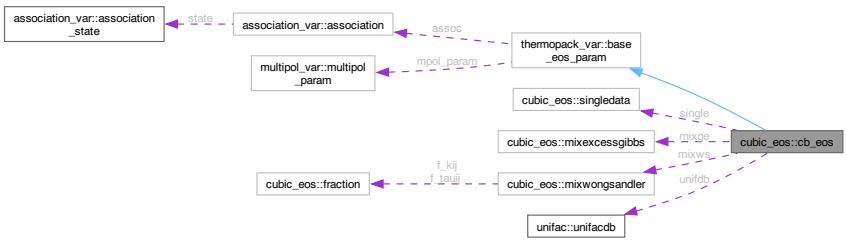
- external.f90

6.22 cubic_eos::cb_eos Type Reference

Inheritance diagram for cubic_eos::cb_eos:



Collaboration diagram for cubic_eos::cb_eos:



Public Member Functions

- procedure, public **dealloc** (eos)
- procedure, public allocate_and_init (eos, nc, eos_label)
- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from thermopack_var::base_eos_param

- procedure(allocate_and_init_intf), deferred, public allocate_and_init (eos, nc, eos_label)
- procedure, public **dealloc** (eos)
- procedure(assign_intf), deferred, pass, public **assign_eos** (this, other)
- generic, public **assignment** assign_eos
- procedure, public **assign_base_eos_param** (this, other)

Public Attributes

- character(len=mix_len) **mruleid**
- character(len=20) **name**
- integer **mruleidx**
- logical **cubic_verbose** = .false.

- real **m1**
- real **m2**
- real **dm1db**
- real **dm1dc**
- real **dm2db**
- real **dm2dc**
- real **d2m1db2**
- real **d2m1dc2**
- real **d2m2db2**
- real **d2m2dc2**
- real **d2m2dbdc**
- real **d2m1dbdc**
- real **delta**
- real **a**
- real **b**
- real **c**

  Dependent of component.
- real **sumn**
- real **suma**

  [Pa∗L^2/mol^2]
- real **sumb**

  Molfraction average of single-component b_i [L/mol].
- real **sumc**

  [L/mol]
- real **pn**
- real **pa**
- real **pb**
- real **pc**
- real **pt**
- real **pv**
- real **ff**
- real **fft**
- real **fftt**
- real **ffn**
- real **ffnv**
- real **ffna**
- real **ffnb**
- real **ffnc**
- real **ffnn**
- real **ffnt**
- real **ffa**
- real **ffaa**
- real **ffab**
- real **ffac**
- real **ffat**
- real **ffb**
- real **ffbb**
- real **ffbc**
- real **ffbt**
- real **ffc**
- real **ffcc**
- real **ffct**
- real **ffv**
- real **ffvt**

- real **ffvv**
- real **ffva**
- real **ffvb**
- real **ffvc**
- real **at**
- real **att**
- real **bt**
- real **btt**
- real, dimension(:), allocatable **ai**
- real, dimension(:), allocatable **ait**
- real, dimension(:,:), allocatable **aij**
- real, dimension(:), allocatable **bi**
- real, dimension(:), allocatable **bit**
- real, dimension(:), allocatable **bitt**
- real, dimension(:), allocatable **ci**
- real, dimension(:,:), allocatable **bij**
- real, dimension(:,:), allocatable **cij**
- integer, dimension(2) **extrm**
- integer, dimension(2) **nextrm**
- integer, dimension(2) **nzfac**
- type(singledata), dimension(:), allocatable **single**
- real, dimension(:,:), allocatable **kij**
- real, dimension(:,:), allocatable **lij**
- real, dimension(:,:), allocatable **lowcase_bij**
- logical **simple_covolmixing**
- type(mixexcessgibbs) **mixge**
- type(mixwongsandler) **mixws**
- type(unifacdb), pointer **unifdb** => NULL()

Public Attributes inherited from thermopack_var::base_eos_param

- character(len=eosid_len) **eosid**

  Eos identifier.
- integer **eosidx**

  Eos group index.
- integer **subeosidx**

  Eos sub-index.
- integer **volumeshiftid** = 0

  0: No volume shift, 1:Peneloux shift
- logical **iselectrolyteeos** = .false.

  Used to enable electrolytes.
- type(association), pointer **assoc** => NULL()
- type(multipol_param), pointer **mpol_param** => NULL()

6.22.1 Member Function/Subroutine Documentation

6.22.1.1 allocate_and_init()

```
procedure, public cubic_eos::cb_eos::allocate_and_init (
             class(cb_eos), intent(inout) eos,
             integer, intent(in) nc,
             character(len=*), intent(in) eos_label )
```

Parameters

in	*eos_label*	EOS label

The documentation for this type was generated from the following file:

- cubic_eos.f90

6.23 cubic::cbbig Type Reference

Public Attributes

- real **a**
- real **b**
- real **c**

The documentation for this type was generated from the following file:

- cubic.f90

6.24 compdata::cidatadb Type Reference

Volume shift parameters.

Public Member Functions

- procedure, public get_vol_trs_c (cid, t, ci, cit, citt, ci_temp_dep)
- procedure, public **set_zero_vol_trs** (cid)

Public Attributes

- character(len=uid_len) **cid**

  The component ID.
- character(len=ref_len) **ref**

  Data group reference.
- character(len=eosid_len) **eosid**

  EOS identifyer.
- character(len=bibref_len) **bib_ref**

  Bibliograpich reference.
- integer **c_type** = VS_CONSTANT

  VS_CONSTANT, VS_LINEAR, VS_QUADRATIC.
- real **cia** = 0

  Volume shift (m3/mol)
- real **cib** = 0

  Volume shift (m3/mol/K)
- real **cic** = 0

  Volume shift (m3/mol/K/K)
- real **cidd** = 0

  Volume shift (m3/mol/K/K/K)
- real **cie** = 0

  Volume shift (m3/mol/K/K/K/K)
- real **cif** = 0

  Volume shift (m3/mol/K/K/K/K/K)

6.24.1 Detailed Description

Volume shift parameters.

6.24.2 Member Function/Subroutine Documentation

6.24.2.1 get_vol_trs_c()

```
procedure, public compdata::cidatadb::get_vol_trs_c (
             class(cidatadb), intent(in) cid,
             real, intent(in) t,
             real, intent(out) ci,
             real, intent(out) cit,
             real, intent(out) citt,
             logical, intent(out) ci_temp_dep )
```

Parameters

in	*t*	Temperature (K)
out	*ci*	Volume translation (m3/mol)
out	*cit*	Volume translation differential (m3/mol/K)
out	*citt*	Volume translation second differential (m3/mol/K2)
out	*ci_temp_dep*	Volume translation is temp. dependent

The documentation for this type was generated from the following file:

- compdata.f90

6.25 hyperdual_mod::cos Interface Reference

Public Member Functions

- elemental type(hyperdual) function **coshyperdual** (v1)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.26 hyperdual_mod::cosh Interface Reference

Public Member Functions

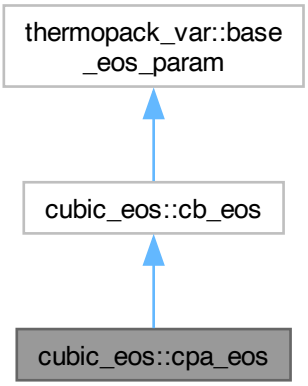
- elemental type(hyperdual) function **coshhyperdual** (v1)

The documentation for this interface was generated from the following file:

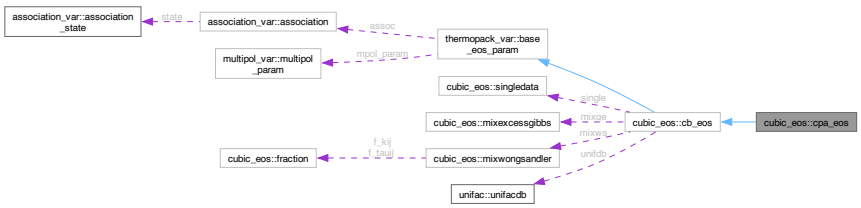
- hyperdual_mod.f90

6.27 cubic_eos::cpa_eos Type Reference

Inheritance diagram for cubic_eos::cpa_eos:



Collaboration diagram for cubic_eos::cpa_eos:



Additional Inherited Members

Public Member Functions inherited from cubic_eos::cb_eos

- procedure, public **dealloc** (eos)
- procedure, public allocate_and_init (eos, nc, eos_label)
- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from thermopack_var::base_eos_param

- procedure(allocate_and_init_intf), deferred, public allocate_and_init (eos, nc, eos_label)
- procedure, public **dealloc** (eos)
- procedure(assign_intf), deferred, pass, public **assign_eos** (this, other)
- generic, public **assignment** assign_eos
- procedure, public **assign_base_eos_param** (this, other)

Public Attributes inherited from cubic_eos::cb_eos

- character(len=mix_len) **mruleid**
- character(len=20) **name**
- integer **mruleidx**

- logical **cubic_verbose** = .false.
- real **m1**
- real **m2**
- real **dm1db**
- real **dm1dc**
- real **dm2db**
- real **dm2dc**
- real **d2m1db2**
- real **d2m1dc2**
- real **d2m2db2**
- real **d2m2dc2**
- real **d2m2dbdc**
- real **d2m1dbdc**
- real **delta**
- real **a**
- real **b**
- real **c**

  Dependent of component.
- real **sumn**
- real **suma**

  [Pa∗L^2/mol^2]
- real **sumb**

  Molfraction average of single-component b_i [L/mol].
- real **sumc**

  [L/mol]
- real **pn**
- real **pa**
- real **pb**
- real **pc**
- real **pt**
- real **pv**
- real **ff**
- real **fft**
- real **fftt**
- real **ffn**
- real **ffnv**
- real **ffna**
- real **ffnb**
- real **ffnc**
- real **ffnn**
- real **ffnt**
- real **ffa**
- real **ffaa**
- real **ffab**
- real **ffac**
- real **ffat**
- real **ffb**
- real **ffbb**
- real **ffbc**
- real **ffbt**
- real **ffc**
- real **ffcc**
- real **ffct**
- real **ffv**

- real **ffvt**
- real **ffvv**
- real **ffva**
- real **ffvb**
- real **ffvc**
- real **at**
- real **att**
- real **bt**
- real **btt**
- real, dimension(:), allocatable **ai**
- real, dimension(:), allocatable **ait**
- real, dimension(:,:), allocatable **aij**
- real, dimension(:), allocatable **bi**
- real, dimension(:), allocatable **bit**
- real, dimension(:), allocatable **bitt**
- real, dimension(:), allocatable **ci**
- real, dimension(:,:), allocatable **bij**
- real, dimension(:,:), allocatable **cij**
- integer, dimension(2) **extrm**
- integer, dimension(2) **nextrm**
- integer, dimension(2) **nzfac**
- type(singledata), dimension(:), allocatable **single**
- real, dimension(:,:), allocatable **kij**
- real, dimension(:,:), allocatable **lij**
- real, dimension(:,:), allocatable **lowcase_bij**
- logical **simple_covolmixing**
- type(mixexcessgibbs) **mixge**
- type(mixwongsandler) **mixws**
- type(unifacdb), pointer **unifdb** => NULL()

Public Attributes inherited from thermopack_var::base_eos_param

- character(len=eosid_len) **eosid**

  Eos identifier.
- integer **eosidx**

  Eos group index.
- integer **subeosidx**

  Eos sub-index.
- integer **volumeshiftid** = 0

  0: No volume shift, 1:Peneloux shift
- logical **iselectrolyteeos** = .false.

  Used to enable electrolytes.
- type(association), pointer **assoc** => NULL()
- type(multipol_param), pointer **mpol_param** => NULL()

The documentation for this type was generated from the following file:

- cubic_eos.f90

6.28 compdata::cpadata Type Reference

Pure component parameters. This data structure stores pure component parameters for the CPA-SRK and CPA-PR equations of state.

Public Attributes

- character(len=eosid_len) **eosid**
- character(len=uid_len) **compname**
- real **a0**

  [Pa∗L^2/mol^2]. The usual a0 parameter.
- real **b**

  [L/mol]. The usual b parameter.
- real, dimension(3) **alphaparams**

  Up to three parameters for use in alpha corr.
- integer **alphacorridx**

  Either cpaClassicIdx, cpaTwuIdx, cpaMcIdx.
- real **eps**

  [J/mol]. Caveat: people sometimes tabulate epsilon/R.
- real **beta**

  [-]
- integer **assoc_scheme**
- character(len=ref_len) **ref**
- character(len=bibref_len) **bib_reference**

6.28.1 Detailed Description

Pure component parameters. This data structure stores pure component parameters for the CPA-SRK and CPA-PR equations of state.
The documentation for this type was generated from the following file:

- compdata.f90

6.29 cubic_eos::cpakijdata Type Reference

Temperature-independent interaction parameters for.

Public Attributes

- character(len=eosid_len) **eosid**
- character(len=uid_len) **uid1**
- character(len=uid_len) **uid2**
- character(len=ref_len) **ref**
- character(len=bibref_len) **bib_ref**
- real **kij_a**
- integer **eps_comb_rule**
- integer **beta_comb_rule**
- real **kij_eps**
- real **kij_beta**

6.29.1 Detailed Description

Temperature-independent interaction parameters for.

- the a parameter in the vdW mixing rules,
- the eps parameter (can be modeled by arithmetic or geometric mean)
- the beta parameter (can be modeled by arithmetic or geometric mean) Geometric mean is numbered 0, arithmetic mean is numbered 1. (It also depends on the scheme used for the two components, but we assume that each component only has one scheme stored in the database.)

The documentation for this type was generated from the following file:

- cubic_eos.f90

6.30 compdata::cpdata Type Reference

Ideal heat capacity at constant pressure.

Public Attributes

- character(len=uid_len) **cid**

  The component ID.
- character(len=ref_len) **ref**

  Data group reference.
- character(len=bibref_len) **bib_ref**
- character(len=bibref_len) **bibliograpich**
- character(len=bibref_len) **reference**
- integer **cptype**
- integer, dimension(10) **correlation**
- integer, dimension(see above) **type**
- real, dimension(10) **cp**
- real, dimension(10) **correlation**
- real, dimension(10) **parameters**
- real **tcpmin**
- real **lower**
- real **temperature**
- real **limit**
- real **k**
- real **tcpmax**
- real **upper**

6.30.1 Detailed Description

Ideal heat capacity at constant pressure.
The documentation for this type was generated from the following file:

- compdata.f90

6.31 eosdata::eos_label_mapping Type Reference

Public Attributes

- integer **eos_idx**
- integer **eos_subidx**
- character(len=short_label_len) **short_label**
- character(len=label_len) **label**
- logical **need_alternative_eos**

The documentation for this type was generated from the following file:

- eosdata.f90

6.32 thermopack_var::eos_param_pointer Type Reference

Public Attributes

- class(base_eos_param), pointer **p_eos**

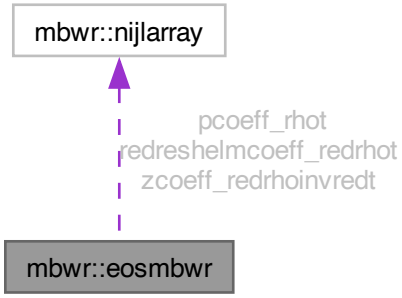
  A trivial type that only contains a pointer to base_eos_param. This type is needed because gfortran does not allow arrays of pointer to base_eos_param, whereas arrays of the eos_param_pointer type is allowed.

The documentation for this type was generated from the following file:

- thermopack_var.f90

6.33 mbwr::eosmbwr Type Reference

MBWR model type for mbwr19 and mbwr32.
Collaboration diagram for mbwr::eosmbwr:



Public Member Functions

- procedure, public **dealloc** (refeosmbwr)

Public Attributes

- character(len=8) **compid**

  Is needed to associate the component to the parameters in MBWRdata.
- character(len=8) **eosid**

  Not used per now.
- character(len=18) **name**

  Not used per now.
- integer **eqno**
- integer **setno**
- integer **lowprop**
- integer **highprop**
- type(nijlarray) **pcoeff_rhot**

  Pressure explicit terms, in the rho-T form. Doesn't take rho∗R∗T-term into account.
- type(nijlarray) **zcoeff_redrhoinvredt**

  Compressibility explicit terms. J: exponent for inverse reduced temperature. I: exponent for reduced density. Doesn't take rho∗R∗T-term into account.
- type(nijlarray) **redreshelmcoeff_redrhot**

  Helmholtz explicit terms, using reduced temperature (J) and reduced density (I). Takes all terms into account.
- integer **ipol**
- integer **iexp**
- integer **bplen**
- integer **belen**
- integer **helmlength**

  Number of terms in the integrated Helmholtz expression.
- integer **helm_poly_len**

  Number of polynomial terms in the integrated Helmholtz expression.
- real **ttriple**

  Triple point temperature.

- real **ptriple**

  Triple point pressure.
- real **tc**

  Critical temperature.
- real **pc**

  Critical pressure.
- real **rc**

  Critical density (mol/L)
- real **zc**

  Critical compressibility factor.
- real **acf**

  Acentric factor, used in the SRK initial value method in the density solver.
- real **gamma**

  The parameter gamma in the MBWR equation.
- real **b_srk**
- real **m_srk**
- real **a0_srk**
- real, dimension(:), allocatable **mbwrparameters**

6.33.1 Detailed Description

MBWR model type for mbwr19 and mbwr32.
The documentation for this type was generated from the following file:

- mbwr.f90

6.34 optimizers::error_function Interface Reference

Public Member Functions

- real function error_function (n, x, nparam, param, of, dofdx, of_old)

6.34.1 Constructor & Destructor Documentation

6.34.1.1 error_function()

```
real function optimizers::error_function::error_function (
            integer, intent(in) n,
            real, dimension(n), intent(in) x,
            integer, intent(in) nparam,
            real, dimension(nparam), intent(in) param,
            real, intent(in) of,
            real, dimension(n), intent(in) dofdx,
            real, intent(in) of_old )  [virtual]
```

Parameters

in	*n*	Dimension of X
in	*x*	Variables
in	*nparam*	Dimension of param
in	*param*	Parameter vector
in	*of*	Objective function value
in	*dofdx*	Differential of objective function
in	*of_old*	Old objective function value

Returns

Calculated error

The documentation for this interface was generated from the following file:

- optimizer.f90

6.35 hyperdual_mod::exp Interface Reference

Public Member Functions

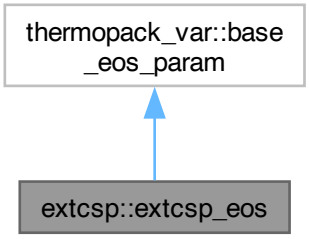
- elemental type(hyperdual) function **exphyperdual** (v1)

The documentation for this interface was generated from the following file:

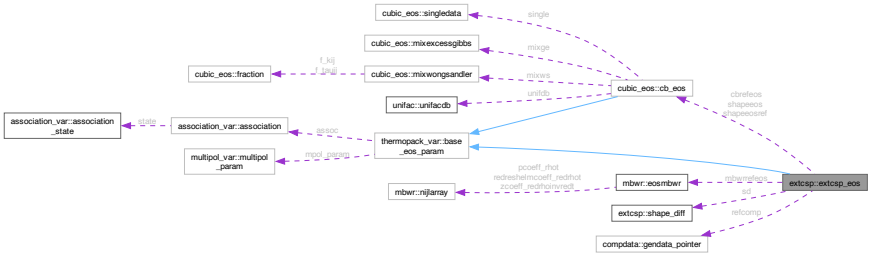
- hyperdual_mod.f90

6.36 extcsp::extcsp_eos Type Reference

Inheritance diagram for extcsp::extcsp_eos:



Collaboration diagram for extcsp::extcsp_eos:



Public Member Functions

- procedure, public **dealloc** (eos)
- procedure, public allocate_and_init (eos, nc, eos_label)
- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from thermopack_var::base_eos_param

- procedure(allocate_and_init_intf), deferred, public allocate_and_init (eos, nc, eos_label)
- procedure, public **dealloc** (eos)
- procedure(assign_intf), deferred, pass, public **assign_eos** (this, other)
- generic, public **assignment** assign_eos
- procedure, public **assign_base_eos_param** (this, other)

Public Attributes

- type(cb_eos) **shapeeos**
- type(cb_eos) **shapeeosref**
- type(cb_eos) **cbrefeos**
- type(eosmbwr), pointer **mbwrrefeos** => NULL()
- class(meos), pointer **nistrefeos** => NULL()
- type(shape_diff) **sd**
- integer **refnc** = 0

  This is set to 1 in subroutine init_component_data_from_db.
- type(gendata_pointer), dimension(:), allocatable **refcomp**

  Will be made to have length refNc=1 in subroutine init_component_data_from_db.
- integer **refeostype**

  Is set to either cubic or mbwr.
- real **tc0**
- real **pc0**

  Critical constants for reference component.
- real **m0**
- real **bc0**
- real **ac0**

  Parameters for the cubic shape eos.

Public Attributes inherited from thermopack_var::base_eos_param

- character(len=eosid_len) **eosid**

  Eos identifier.
- integer **eosidx**

  Eos group index.
- integer **subeosidx**

  Eos sub-index.
- integer **volumeshiftid** = 0

  0: No volume shift, 1:Peneloux shift
- logical **iselectrolyteeos** = .false.

  Used to enable electrolytes.
- type(association), pointer **assoc** => NULL()
- type(multipol_param), pointer **mpol_param** => NULL()

6.36.1 Member Function/Subroutine Documentation

6.36.1.1 allocate_and_init()

```
procedure, public extcsp::extcsp_eos::allocate_and_init (
             class(extcsp_eos), intent(inout) eos,
             integer, intent(in) nc,
             character(len=*), intent(in) eos_label )
```

Parameters

in	*nc*	Number of components
in	*eos_label*	EOS and component label

The documentation for this type was generated from the following file:

- extcsp.f90

6.37 cubic_eos::fraction Type Reference

Public Attributes

- real, dimension(ndegreepoly+1) **pnum**

  Numerator polynom.
- real, dimension(ndegreepoly+1) pden

  Denominator polynom.

6.37.1 Member Data Documentation

6.37.1.1 pden

```
real, dimension(ndegreepoly+1) cubic_eos::fraction::pden
```

Denominator polynom.
y = (pNum(1)+pNum(2)∗x+...pNum(n-1)∗x∗∗n)/(pDen(1)+pDen(2)∗x+...pDen(n-1)∗x∗∗n) where n = nDegreePoly.
Generally pDen(1) = 1.0
The documentation for this type was generated from the following file:

- cubic_eos.f90

6.38 nonlinear_solvers::function_template Interface Reference

Public Member Functions

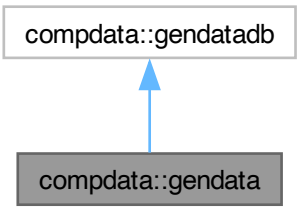
- subroutine **function_template** (f, x, p)

The documentation for this interface was generated from the following file:

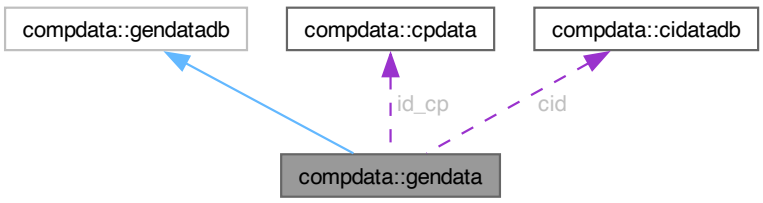
- nonlinear_solvers.f90

6.39 compdata::gendata Type Reference

Inheritance diagram for compdata::gendata:



Collaboration diagram for compdata::gendata:



Public Member Functions

- procedure, public **init_from_name** gendata_init_from_name
- procedure, pass, public **assign_comp** (this, cmp)

  Assignment operator for gendata.

Public Member Functions inherited from compdata::gendatadb

- procedure, pass, public **assign_comp** (this, cmp)

  Assignment operator for gendatadb.
- generic, public **assignment** assign_comp

Public Attributes

- type(cpdata) **id_cp**

  Ideal gas Cp correlation.
- type(cidatadb) **cid**

  Volume shift parameters.
- integer **assoc_scheme**

  Association scheme for use in the SAFT model. The various schemes are defined in saft_parameters_db.f90.

Public Attributes inherited from compdata::gendatadb

- character(len=uid_len) **ident**

  The component ID.
- character(len=formula_len) **formula**

  Chemical formula.
- character(len=comp_name_len) **name**

  The component name.
- real **mw**

  Mole weight[g/mol].
- real **tc**

  Critical temperature [K].
- real **pc**

  Critical pressure [Pa].
- real **zc**

  Critical compressibility [-].
- real **acf**

  Acentric factor [-].

- real **tb**

  Normal boiling point [K].
- real **ttr**

  Triple point temperature [K].
- real **ptr**

  Triple point temperature [K].
- real **sref**

  Reference entropy [J/mol/K].
- real **href**

  Reference enthalpy [J/mol].
- real **dfh**

  Enthalpy of formation [J/mol].
- real **dfg**

  Gibbs energy of formation [J/mol].
- integer **psatcode**

  Vapour pressure correlation 1: Antoine 2: Wilson (Michelsen) 3: Starling.
- real, dimension(3) **ant**

  Vapour pressure correlation parameters.
- real **tantmin**

  Vapour pressure correlation lower temperature limit [K].
- real **tantmax**

  Vapour pressure correlation upper temperature limit [K].
- real **zra**

  Rackett compressibility factor.
- real **mu_dipole**

  Electric dipole moment (D)
- real **q_quadrupole**

  Electric quadrupole moment (ÃåD)

The documentation for this type was generated from the following file:

- compdata.f90

6.40 compdata::gendata_pointer Type Reference

Public Attributes

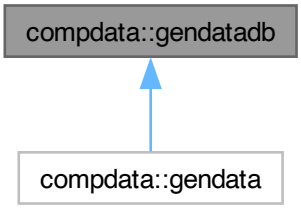
- class(gendata), pointer **p_comp** => NULL()

The documentation for this type was generated from the following file:

- compdata.f90

6.41 compdata::gendatadb Type Reference

Inheritance diagram for compdata::gendatadb:



Public Member Functions

- procedure, pass, public **assign_comp** (this, cmp)

  Assignment operator for gendatadb.
- generic, public **assignment** assign_comp

Public Attributes

- character(len=uid_len) **ident**

  The component ID.
- character(len=formula_len) **formula**

  Chemical formula.
- character(len=comp_name_len) **name**

  The component name.
- real **mw**

  Mole weight[g/mol].
- real **tc**

  Critical temperature [K].
- real **pc**

  Critical pressure [Pa].
- real **zc**

  Critical compressibility [-].
- real **acf**

  Acentric factor [-].
- real **tb**

  Normal boiling point [K].
- real **ttr**

  Triple point temperature [K].
- real **ptr**

  Triple point temperature [K].
- real **sref**

  Reference entropy [J/mol/K].
- real **href**

  Reference enthalpy [J/mol].
- real **dfh**

Enthalpy of formation [J/mol].

- real **dfg**

  Gibbs energy of formation [J/mol].

- integer **psatcode**

  Vapour pressure correlation 1: Antoine 2: Wilson (Michelsen) 3: Starling.

- real, dimension(3) **ant**

  Vapour pressure correlation parameters.

- real **tantmin**

  Vapour pressure correlation lower temperature limit [K].

- real **tantmax**

  Vapour pressure correlation upper temperature limit [K].

- real **zra**

  Rackett compressibility factor.

- real **mu_dipole**

  Electric dipole moment (D)

- real **q_quadrupole**

  Electric quadrupole moment (ÃåD)

The documentation for this type was generated from the following file:

- compdata.f90

6.42 gergmixdb::gerg_mix_data Type Reference

Public Attributes

- character(len=uid_len) **ident1**

  The component ID.

- character(len=uid_len) **ident2**

  The component ID.

- real **fij**

  Departure function parameter.

- integer **num_mix**

  Number of parameters.

- real, dimension(12) **n_mix**
- real, dimension(12) **t_mix**
- integer, dimension(12) **d_mix**
- real, dimension(12) **eta_mix**
- real, dimension(12) **gamma_mix**
- real, dimension(12) **epsilon_mix**
- real, dimension(12) **beta_mix**
- integer **num_exp**

  Number of exponential terms.

The documentation for this type was generated from the following file:

- gergmixdb.f90

6.43 gergmixdb::gerg_mix_reducing Type Reference

Public Attributes

- character(len=uid_len) **ident1**

  The component ID.
- character(len=uid_len) **ident2**

  The component ID.
- real **beta_v**

  Reducing density parameter.
- real **gamma_v**

  Reducing density parameter.
- real **beta_t**

  Reducing temperature parameter.
- real **gamma_t**

  Reducing temperature parameter.

The documentation for this type was generated from the following file:

- gergmixdb.f90

6.44 gergdatadb::gergdata Type Reference

Public Attributes

- character(len=uid_len) **ident**

  The component ID.
- character(len=comp_name_len) **name**

  The component name.
- real **mw**

  Mole weight (g/mol)
- real **ttr**

  Triple point temperature (K)
- real **ptr**

  Triple point pressure (kPa)
- real **tc**

  Critical temperature (K)
- real **pc**

  Critical pressure (kPa)
- real **rhoc**

  Critical density (mol/l)
- real **tr**

  Reducing temperature (K)
- real **rhor**

  Reducing density (mol/l)
- real **rgas**

  Gas constant (J/mol-K)
- real **acf**

  Acentric factor.
- real **t_max**

  Reducing temperature (K)
- real **p_max**

  Reducing temperature (kPa)

- integer **n_eos**
- real, dimension(24) **a_eos**
- real, dimension(24) **t_eos**
- integer, dimension(24) **d_eos**
- integer, dimension(24) **l_eos**
- integer **n_cosh**
- integer **n_sinh**
- real, dimension(7) **n_id**
- real, dimension(7) **t_id**

The documentation for this type was generated from the following file:

- gergdatadb.f90

6.45 idealh2::h2func Interface Reference

Public Member Functions

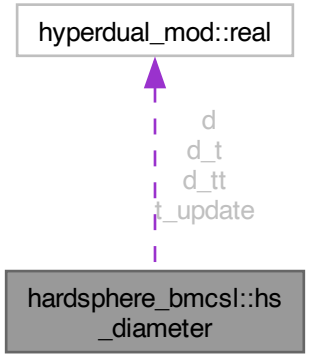
- integer function **geth2index** (ident)

The documentation for this interface was generated from the following file:

- idealh2.f90

6.46 hardsphere_bmcsl::hs_diameter Type Reference

Container for temperature dependent hard-sphere diameter and differentials.
Collaboration diagram for hardsphere_bmcsl::hs_diameter:



Public Member Functions

- procedure, public **allocate** (dhs, nc)

  Allocated hs_diameter memory.
- procedure, public **deallocate** (dhs)

  Free allocated hs_diameter memory.
- procedure, public **assign_hs_diameter** (this, other)
- generic, public **assignment** (this, other)

Public Attributes

- real, dimension(:), allocatable **d**

  Hard sphere diameter.
- real, dimension(:), allocatable **d_t**

  Temperature differential of hard sphere diameter.
- real, dimension(:), allocatable **d_tt**

  Second temperature differential of hard sphere diameter.
- real **t_update** = 0

  d calculated for T:

6.46.1 Detailed Description

Container for temperature dependent hard-sphere diameter and differentials.
The documentation for this type was generated from the following file:

- hardsphere_bmcsl.f90

6.47 hyperdual_mod::hyperdual Type Reference

Derived type for hyperdual numbers.

Public Attributes

- real(dp) **f0** = 0

  real part of the hyperdual number
- real(dp) **f1** = 0

  ε_1*-part of the hyperdual number*
- real(dp) **f2** = 0

  ε_2*-part of the hyperdual number*
- real(dp) **f3** = 0

  ε_3*-part of the hyperdual number*
- real(dp) **f12** = 0

  $\varepsilon_1\varepsilon_2$*-part of the*
- real(dp) **f13** = 0

  $\varepsilon_1\varepsilon_3$*-part of the*
- real(dp) **f23** = 0

  $\varepsilon_2\varepsilon_3$*-part of the*
- real(dp) **f123** = 0

  $\varepsilon_1\varepsilon_2\varepsilon_3$*-part of the*
- integer **order** = 2

  Overall order of differential. Defaults to 2. Order 3 must be activated.

6.47.1 Detailed Description

Derived type for hyperdual numbers.
Hyperdual numbers are represented by the tuple $\mathbf{f} = [f_0, f_1, f_2, f_3, f_{12}, f_{13}, f_{23}, f_{123}] = f_0 + f_1\varepsilon_1 + f_2\varepsilon_2 + f_3\varepsilon_3 + f_{12}\varepsilon_1\varepsilon_2 + f_{13}\varepsilon_1\varepsilon_3 + f_{23}\varepsilon_2\varepsilon_3 + f_{123}\varepsilon_1\varepsilon_2\varepsilon_3$. Calculations specificaions are defined in module hyperdual_mod.
The documentation for this type was generated from the following file:

- hyperdual_mod.f90

6.48 hyperdual_utility::hyperdual_fres Interface Reference

Public Member Functions

- type(hyperdual) function **hyperdual_fres** (p_eos, nc, t, v, n)

The documentation for this interface was generated from the following file:

- hyperdual_utility.f90

6.49 multiparameter_base::init_intf Interface Reference

Public Member Functions

- subroutine **init_intf** (this, use_rgas_fit)

The documentation for this interface was generated from the following file:

- multiparameter_base.f90

6.50 hyperdual_mod::int Interface Reference

Public Member Functions

- elemental integer function **inthyperdual** (v1)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.51 cubic_eos::intergedatadb Type Reference

Public Attributes

- character(len=eosid_len) **eosid**
- character(len=eosid_len) **mruleid**
- character(len=ref_len) **ref**
- character(len=bibref_len) **bib_ref**
- character(len=uid_len) **uid1**
- character(len=uid_len) **uid2**
- real **kijvalue**
- integer **correlation**
- real, dimension(2) **alphaijvalue**
- real, dimension(3) **polyij**
- real, dimension(3) **polyji**

The documentation for this type was generated from the following file:

- cubic_eos.f90

6.52 nonlinear_solvers::jacobian_template Interface Reference

Public Member Functions

- subroutine **jacobian_template** (j, x, p)

The documentation for this interface was generated from the following file:

- nonlinear_solvers.f90

6.53 cubic_eos::kijdatadb Type Reference

Public Attributes

- character(len=eosid_len) **eosid**
- character(len=eosid_len) **mruleid**
- character(len=ref_len) **ref**
- character(len=bibref_len) **bib_ref**
- character(len=uid_len) **uid1**
- character(len=uid_len) **uid2**
- real **kijvalue**

The documentation for this type was generated from the following file:

- cubic_eos.f90

6.54 cubic_eos::lijdatadb Type Reference

Public Attributes

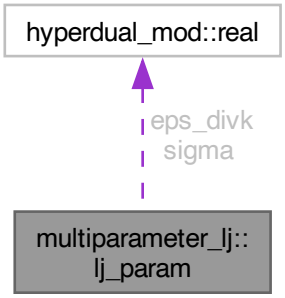
- character(len=eosid_len) **eosid**
- character(len=eosid_len) **mruleid**
- character(len=ref_len) **ref**
- character(len=bibref_len) **bib_ref**
- character(len=uid_len) **uid1**
- character(len=uid_len) **uid2**
- real **lijvalue**

The documentation for this type was generated from the following file:

- cubic_eos.f90

6.55 multiparameter_lj::lj_param Type Reference

Noble gas parameters for LJ EOS.
Collaboration diagram for multiparameter_lj::lj_param:



Public Attributes

- character(len=20) **comp_name**
- real **eps_divk**
- real **sigma**

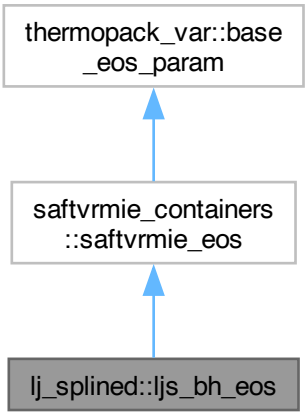
6.55.1 Detailed Description

Noble gas parameters for LJ EOS.
The documentation for this type was generated from the following file:

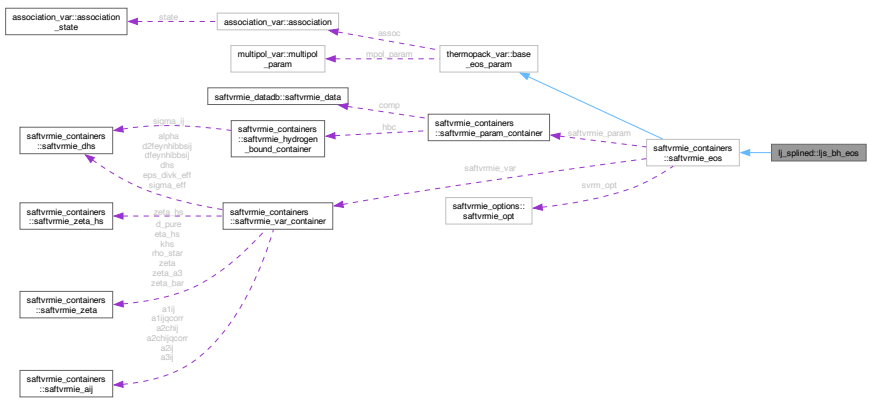
- multiparameter_lj.f90

6.56 lj_splined::ljs_bh_eos Type Reference

Inheritance diagram for lj_splined::ljs_bh_eos:



Collaboration diagram for lj_splined::ljs_bh_eos:



Public Member Functions

- procedure, public **set_sigma_eps** (ljs, sigma, eps_depth_divk)
- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from saftvrmie_containers::saftvrmie_eos

- procedure, public **dealloc** (eos)

- procedure, public allocate_and_init (eos, nc, eos_label)
- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from thermopack_var::base_eos_param

- procedure(allocate_and_init_intf), deferred, public allocate_and_init (eos, nc, eos_label)
- procedure, public **dealloc** (eos)
- procedure(assign_intf), deferred, pass, public **assign_eos** (this, other)
- generic, public **assignment** assign_eos
- procedure, public **assign_base_eos_param** (this, other)

Public Attributes

- logical **use_lafitte_a3** = .false.
- logical **enable_chi_correction** = .true.
- logical **enable_hs** = .true.
- logical **enable_a1** = .true.
- logical **enable_a2** = .true.
- logical **enable_a3** = .true.

Public Attributes inherited from saftvrmie_containers::saftvrmie_eos

- logical **ovner_of_saftvrmie_param** = .false.
- type(saftvrmie_param_container), pointer **saftvrmie_param** => NULL()
- type(saftvrmie_var_container), pointer **saftvrmie_var** => NULL()
- logical **ovner_of_svrm_opt** = .false.
- type(saftvrmie_opt), pointer **svrm_opt** => NULL()

Public Attributes inherited from thermopack_var::base_eos_param

- character(len=eosid_len) **eosid**

  Eos identifier.
- integer **eosidx**

  Eos group index.
- integer **subeosidx**

  Eos sub-index.
- integer **volumeshiftid** = 0

  0: No volume shift, 1:Peneloux shift
- logical **iselectrolyteeos** = .false.

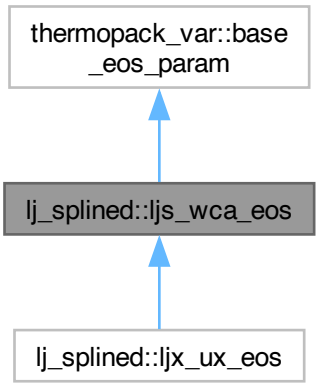
  Used to enable electrolytes.
- type(association), pointer **assoc** => NULL()
- type(multipol_param), pointer **mpol_param** => NULL()

The documentation for this type was generated from the following file:

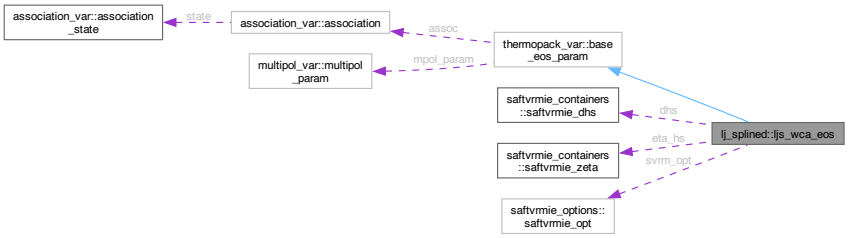
- lj_splined.f90

6.57 lj_splined::ljs_wca_eos Type Reference

Inheritance diagram for lj_splined::ljs_wca_eos:



Collaboration diagram for lj_splined::ljs_wca_eos:



Public Member Functions

- procedure, public **set_sigma_eps** (ljs, sigma, eps_depth_divk)
- procedure, public **dealloc** (eos)
- procedure, public allocate_and_init (eos, nc, eos_label)
- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from thermopack_var::base_eos_param

- procedure(allocate_and_init_intf), deferred, public allocate_and_init (eos, nc, eos_label)
- procedure, public **dealloc** (eos)
- procedure(assign_intf), deferred, pass, public **assign_eos** (this, other)
- generic, public **assignment** assign_eos
- procedure, public **assign_base_eos_param** (this, other)

Public Attributes

- real **sigma**
- real **eps_divk**
- type(saftvrmie_dhs) **dhs**

The hard-sphere diameter.

- type(saftvrmie_zeta) **eta_hs**

  Packing fraction.
- logical **enable_cavity** = .true.
- logical **enable_hs** = .true.
- logical **enable_a1** = .true.
- logical **enable_a2** = .true.
- logical **enable_a3** = .true.
- logical **enable_a4** = .true.
- logical **ovner_of_svrm_opt** = .false.
- type(saftvrmie_opt), pointer **svrm_opt** => NULL()

Public Attributes inherited from thermopack_var::base_eos_param

- character(len=eosid_len) **eosid**

  Eos identifier.
- integer **eosidx**

  Eos group index.
- integer **subeosidx**

  Eos sub-index.
- integer **volumeshiftid** = 0

  0: No volume shift, 1:Peneloux shift
- logical **iselectrolyteeos** = .false.

  Used to enable electrolytes.
- type(association), pointer **assoc** => NULL()
- type(multipol_param), pointer **mpol_param** => NULL()

6.57.1 Member Function/Subroutine Documentation

6.57.1.1 allocate_and_init()

```
procedure, public lj_splined::ljs_wca_eos::allocate_and_init (
             class(ljs_wca_eos), intent(inout) eos,
             integer, intent(in) nc,
             character(len=*), intent(in) eos_label )
```

Parameters

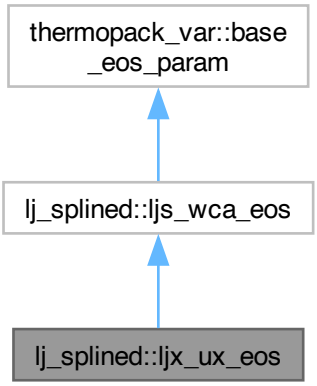
in	*eos_label*	EOS label

The documentation for this type was generated from the following file:

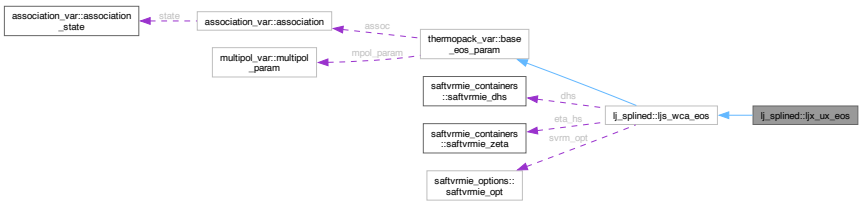
- lj_splined.f90

6.58 lj_splined::ljx_ux_eos Type Reference

Inheritance diagram for lj_splined::ljx_ux_eos:



Collaboration diagram for lj_splined::ljx_ux_eos:



Public Member Functions

- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from lj_splined::ljs_wca_eos

- procedure, public **set_sigma_eps** (ljs, sigma, eps_depth_divk)
- procedure, public **dealloc** (eos)
- procedure, public allocate_and_init (eos, nc, eos_label)
- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from thermopack_var::base_eos_param

- procedure(allocate_and_init_intf), deferred, public allocate_and_init (eos, nc, eos_label)
- procedure, public **dealloc** (eos)
- procedure(assign_intf), deferred, pass, public **assign_eos** (this, other)
- generic, public **assignment** assign_eos
- procedure, public **assign_base_eos_param** (this, other)

Public Attributes

- logical **lj_potential**
- logical **is_uf_theory**
- logical **enable_virial_term** = .true.
- logical **use_temperature_dependent_u_fraction** = .false.

Public Attributes inherited from lj_splined::ljs_wca_eos

- real **sigma**
- real **eps_divk**
- type(saftvrmie_dhs) **dhs**

  The hard-sphere diameter.
- type(saftvrmie_zeta) **eta_hs**

  Packing fraction.
- logical **enable_cavity** = .true.
- logical **enable_hs** = .true.
- logical **enable_a1** = .true.
- logical **enable_a2** = .true.
- logical **enable_a3** = .true.
- logical **enable_a4** = .true.
- logical **ovner_of_svrm_opt** = .false.
- type(saftvrmie_opt), pointer **svrm_opt** => NULL()

Public Attributes inherited from thermopack_var::base_eos_param

- character(len=eosid_len) **eosid**

  Eos identifier.
- integer **eosidx**

  Eos group index.
- integer **subeosidx**

  Eos sub-index.
- integer **volumeshiftid** = 0

  0: No volume shift, 1:Peneloux shift
- logical **iselectrolyteeos** = .false.

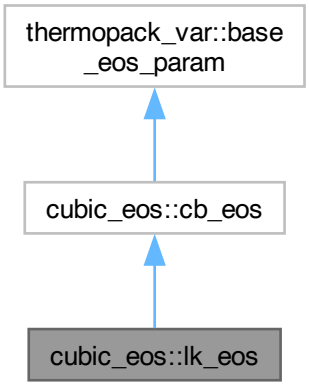
  Used to enable electrolytes.
- type(association), pointer **assoc** => NULL()
- type(multipol_param), pointer **mpol_param** => NULL()

The documentation for this type was generated from the following file:

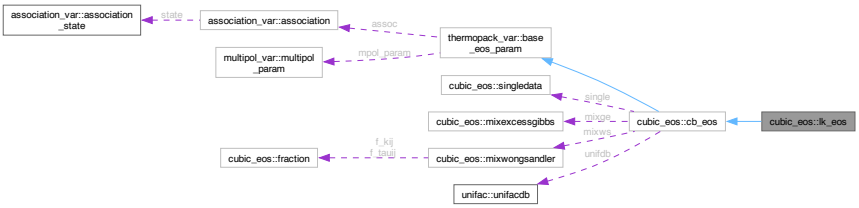
- lj_splined.f90

6.59 cubic_eos::lk_eos Type Reference

Inheritance diagram for cubic_eos::lk_eos:



Collaboration diagram for cubic_eos::lk_eos:



Additional Inherited Members

Public Member Functions inherited from cubic_eos::cb_eos

- procedure, public **dealloc** (eos)
- procedure, public allocate_and_init (eos, nc, eos_label)
- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from thermopack_var::base_eos_param

- procedure(allocate_and_init_intf), deferred, public allocate_and_init (eos, nc, eos_label)
- procedure, public **dealloc** (eos)
- procedure(assign_intf), deferred, pass, public **assign_eos** (this, other)
- generic, public **assignment** assign_eos
- procedure, public **assign_base_eos_param** (this, other)

Public Attributes inherited from cubic_eos::cb_eos

- character(len=mix_len) **mruleid**
- character(len=20) **name**
- integer **mruleidx**

- logical **cubic_verbose** = .false.
- real **m1**
- real **m2**
- real **dm1db**
- real **dm1dc**
- real **dm2db**
- real **dm2dc**
- real **d2m1db2**
- real **d2m1dc2**
- real **d2m2db2**
- real **d2m2dc2**
- real **d2m2dbdc**
- real **d2m1dbdc**
- real **delta**
- real **a**
- real **b**
- real **c**

  Dependent of component.
- real **sumn**
- real **suma**

  [Pa∗L^2/mol^2]
- real **sumb**

  Molfraction average of single-component b_i [L/mol].
- real **sumc**

  [L/mol]
- real **pn**
- real **pa**
- real **pb**
- real **pc**
- real **pt**
- real **pv**
- real **ff**
- real **fft**
- real **fftt**
- real **ffn**
- real **ffnv**
- real **ffna**
- real **ffnb**
- real **ffnc**
- real **ffnn**
- real **ffnt**
- real **ffa**
- real **ffaa**
- real **ffab**
- real **ffac**
- real **ffat**
- real **ffb**
- real **ffbb**
- real **ffbc**
- real **ffbt**
- real **ffc**
- real **ffcc**
- real **ffct**
- real **ffv**

- real **ffvt**
- real **ffvv**
- real **ffva**
- real **ffvb**
- real **ffvc**
- real **at**
- real **att**
- real **bt**
- real **btt**
- real, dimension(:), allocatable **ai**
- real, dimension(:), allocatable **ait**
- real, dimension(:,:), allocatable **aij**
- real, dimension(:), allocatable **bi**
- real, dimension(:), allocatable **bit**
- real, dimension(:), allocatable **bitt**
- real, dimension(:), allocatable **ci**
- real, dimension(:,:), allocatable **bij**
- real, dimension(:,:), allocatable **cij**
- integer, dimension(2) **extrm**
- integer, dimension(2) **nextrm**
- integer, dimension(2) **nzfac**
- type(singledata), dimension(:), allocatable **single**
- real, dimension(:,:), allocatable **kij**
- real, dimension(:,:), allocatable **lij**
- real, dimension(:,:), allocatable **lowcase_bij**
- logical **simple_covolmixing**
- type(mixexcessgibbs) **mixge**
- type(mixwongsandler) **mixws**
- type(unifacdb), pointer **unifdb** => NULL()

Public Attributes inherited from thermopack_var::base_eos_param

- character(len=eosid_len) **eosid**

  Eos identifier.
- integer **eosidx**

  Eos group index.
- integer **subeosidx**

  Eos sub-index.
- integer **volumeshiftid** = 0

  0: No volume shift, 1:Peneloux shift
- logical **iselectrolyteeos** = .false.

  Used to enable electrolytes.
- type(association), pointer **assoc** => NULL()
- type(multipol_param), pointer **mpol_param** => NULL()

The documentation for this type was generated from the following file:

- cubic_eos.f90

6.60 hyperdual_mod::log Interface Reference

Public Member Functions

- elemental type(hyperdual) function **loghyperdual** (v1)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.61 hyperdual_mod::log10 Interface Reference

Public Member Functions

- elemental type(hyperdual) function **log10hyperdual** (v1)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.62 hyperdual_mod::max Interface Reference

Public Member Functions

- elemental type(hyperdual) function **max_dd** (v1, v2)
- elemental type(hyperdual) function **max_ddd** (v1, v2, v3)
- elemental type(hyperdual) function **max_dr** (v1, v2)
- elemental type(hyperdual) function **max_rd** (v1, v2)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.63 mbwrdata::mbwr19data Type Reference

Public Attributes

- integer **eqno**
- character(len=10) **comid**
- integer **setno**
- integer **lowprop**
- integer **highprop**
- integer **ndata**
- real ∗8, dimension(20) **coeff**

The documentation for this type was generated from the following file:

- mbwrdata.f90

6.64 mbwrdata::mbwr32data Type Reference

Public Attributes

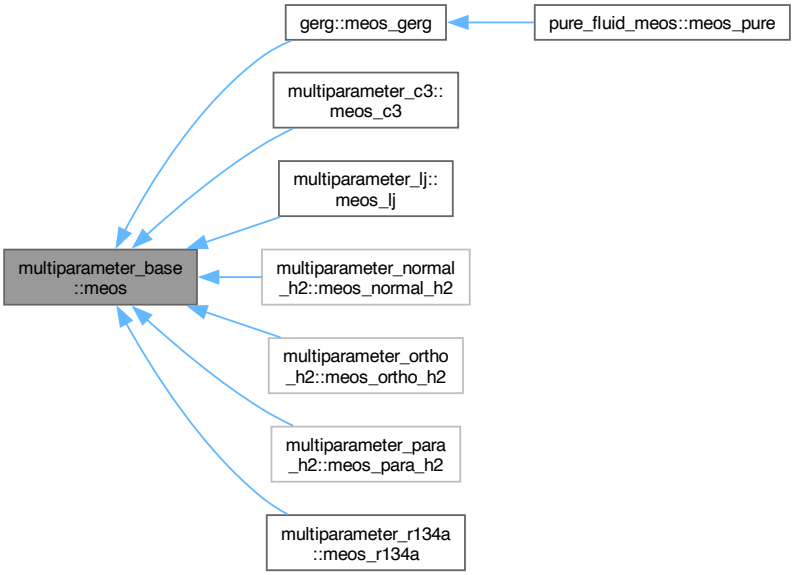
- integer **eqno**
- character(len=10) **comid**
- integer **setno**
- integer **lowprop**
- integer **highprop**
- integer **ndata**
- real ∗8, dimension(33) **coeff**

The documentation for this type was generated from the following file:

- mbwrdata.f90

6.65 multiparameter_base::meos Type Reference

Base class for multiparameter equations of state.
Inheritance diagram for multiparameter_base::meos:



Public Member Functions

- procedure, public **mp_pressure** (this, rho, t, p, p_rho, p_t)

  Pressure and (optionally) its derivatives.
- procedure, public **alpha_to_f_conversion** (this, t, v, n, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- procedure, public **calc_f** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- procedure, public **calc_fid** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)

  Calculate reduced ideal Helmholtz energy and differentials.
- procedure, public **calc_zfac** (this, t, p, n, phase, z, z_t, z_p, z_n)
- procedure, public **calc_lnphi** (this, t, p, n, phase, lnphi, lnphi_t, lnphi_p, lnphi_n)
- procedure, public **calc_entropy** (this, t, p, n, phase, s, s_t, s_p, s_n, residual)
- procedure, public **calc_enthalpy** (this, t, p, n, phase, h, h_t, h_p, h_n, residual)
- procedure, public **calc_resgibbs** (this, t, p, n, phase, g, g_t, g_p, g_n)
- procedure, public densitysolver (this, t_spec, p_spec, phase_spec, rho, phase_found, ierr)
- procedure, public alphaderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, residual)
- procedure, public alpharesderivs_tv (this, t, v, alpr, alpr_t, alpr_v, alpr_tt, alpr_tv, alpr_vv, alpr_n, alpr_tn, alpr_vn, alpr_nn)
- procedure, public alphaidderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, alp_n, alp_tn, alp_vn, alp_nn)
- procedure, public **getcritpoint** (this, tcrit, pcrit, rhocrit)
- procedure(init_intf), deferred, public **init** (this, use_rgas_fit)

  Initiate compName, critical point and triple point.
- procedure, public cv (this, t, v)

  Isochoric heat capacity.
- procedure, public cp (this, t, v)

  Isobaric heat capacity.

- procedure, public speed_of_sound (this, t, v)

  [m/s]
- procedure, public **get_ref_state_spec** (this, ref_state, t, p, phase, solve)

  Get specification for the current reference state.
- procedure, public **set_ref_state** (this, t, p, v, h, s)

  Set reference state.
- procedure(satdeltaestimate_intf), deferred, public satdeltaestimate (this, tau, phase)

  An estimate delta_sat(tau_sat) for use in density solver.
- procedure(alpha0derivs_intf), deferred, public alpha0derivs_taudelta (this, delta, tau, alp0)

  *[d^{j}alpha0/(d_tau)^j]*tau^j*
- procedure(alpharesderivs_intf), deferred, public alpharesderivs_taudelta (this, delta, tau, alpr)

  *[d^{i+j}alphaRes/(d_delta)^i(d_tau)^j]*delta^i*tau^j*
- procedure(alpha0_hd_intf), deferred, public alpha0_hd_taudelta (this, delta, tau)

  Calculate alpha0 using hyperdual numbers.
- procedure(alphares_hd_intf), deferred, public alphares_hd_taudelta (this, delta, tau)

  Calculate alphaRes using hyperdual numbers.
- procedure, public **assign_meos_base** (this, other)
- procedure(assign_meos_intf), deferred, pass, public **assign_meos** (this, other)
- generic, public **assignment** assign_meos

Public Attributes

- character(len=20), public **compname**

  Parameters in SI units. These are set in the deferred init routine.
- real, public **tc**
- real, public **pc**
- real, public **rc**
- real, public **acf**
- real, public **t_triple**
- real, public **p_triple**
- real, public **rholiq_triple**
- real, public **rhovap_triple**
- real, public **molarmass**

  (kg/mol)
- real, public **maxt**
- real, public **maxp**

  (K), (Pa)
- real, public **rgas_meos** = Rgas_default
- real, public **rgas_fit**

6.65.1 Detailed Description

Base class for multiparameter equations of state.

6.65.2 Member Function/Subroutine Documentation

6.65.2.1 alpha0_hd_taudelta()

```
procedure(alpha0_hd_intf), deferred, public multiparameter_base::meos::alpha0_hd_taudelta (
            class(meos) this,
            type(hyperdual), intent(in) delta,
            type(hyperdual), intent(in) tau )  [pure virtual]
```

Calculate alpha0 using hyperdual numbers.

Parameters

in	*delta*	Reduced density (-)
in	*tau*	Reduced temperature (-)

Returns

Ideal reduced Helmholtz energy

6.65.2.2 alpha0derivs_taudelta()

```
procedure(alpha0derivs_intf), deferred, public multiparameter_base::meos::alpha0derivs_↩
taudelta (
             class(meos) this,
             real, intent(in) delta,
             real, intent(in) tau,
             real, dimension(0:2,0:2), intent(out) alp0 )  [pure virtual]
```

[d^{j}alpha0/(d_tau)^j]∗tau^j

Parameters

in	*delta*	Reduced density (-)
in	*tau*	Reduced temperature (-)
out	*alp0*	alp0(i,j) = [(d_delta)^i(d_tau)^j alpha0]∗delta^i∗tau^j

6.65.2.3 alphaderivs_tv()

```
procedure, public multiparameter_base::meos::alphaderivs_tv (
             class(meos) this,
             real, intent(in) t,
             real, intent(in) v,
             real, intent(out), optional alp,
             real, intent(out), optional alp_t,
             real, intent(out), optional alp_v,
             real, intent(out), optional alp_tt,
             real, intent(out), optional alp_tv,
             real, intent(out), optional alp_vv,
             logical, intent(in), optional residual )
```

Parameters

	this	Calling class
in	*v*	Temperature (K) and molar volume (m^3/mol)
out	*alp*	A/(nRT)

6.65.2.4 alphaidderivs_tv()

```
procedure, public multiparameter_base::meos::alphaidderivs_tv (
             class(meos) this,
             real, intent(in) t,
             real, intent(in) v,
             real, intent(out), optional alp,
             real, intent(out), optional alp_t,
             real, intent(out), optional alp_v,
```

```
            real, intent(out), optional alp_tt,
            real, intent(out), optional alp_tv,
            real, intent(out), optional alp_vv,
            real, intent(out), optional alp_n,
            real, intent(out), optional alp_tn,
            real, intent(out), optional alp_vn,
            real, intent(out), optional alp_nn )
```

Parameters

	this	Calling class
in	*v*	Temperature (K) and molar volume (m^3/mol)
out	*alp*	A/(nRT)

6.65.2.5 alphares_hd_taudelta()

```
procedure(alphares_hd_intf), deferred, public multiparameter_base::meos::alphares_hd_taudelta
(
            class(meos) this,
            type(hyperdual), intent(in) delta,
            type(hyperdual), intent(in) tau )  [pure virtual]
```

Calculate alphaRes using hyperdual numbers.

Parameters

in	*delta*	Reduced density (-)
in	*tau*	Reduced temperature (-)

Returns

Residual reduced Helmholtz energy

6.65.2.6 alpharesderivs_taudelta()

```
procedure(alpharesderivs_intf), deferred, public multiparameter_base::meos::alpharesderivs_↩
taudelta (
            class(meos) this,
            real, intent(in) delta,
            real, intent(in) tau,
            real, dimension(0:2,0:2), intent(out) alpr )  [pure virtual]
```

[d^{i+j}alphaRes/(d_delta)^i(d_tau)^j]*delta^i*tau^j

Parameters

in	*delta*	Reduced density (-)
in	*tau*	Reduced temperature (-)
out	*alpr*	alpr(i,j) = [(d_delta)^i(d_tau)^j alphaRes]*delta^i*tau^j

6.65.2.7 alpharesderivs_tv()

```
procedure, public multiparameter_base::meos::alpharesderivs_tv (
            class(meos) this,
            real, intent(in) t,
            real, intent(in) v,
```

```
            real, intent(out), optional alpr,
            real, intent(out), optional alpr_t,
            real, intent(out), optional alpr_v,
            real, intent(out), optional alpr_tt,
            real, intent(out), optional alpr_tv,
            real, intent(out), optional alpr_vv,
            real, intent(out), optional alpr_n,
            real, intent(out), optional alpr_tn,
            real, intent(out), optional alpr_vn,
            real, intent(out), optional alpr_nn )
```

Parameters

	this	Calling class
in	*v*	Temperature (K) and molar volume (m^3/mol)
out	*alpr*	A/(nRT)

6.65.2.8 cp()

```
procedure, public multiparameter_base::meos::cp (
            class(meos) this,
            real, intent(in) t,
            real, intent(in) v )
```

Isobaric heat capacity.

Parameters

	this	Calling class
in	*v*	Temperature (K) and molar volume (m^3/mol)

Returns

[J/(mol∗K)]

6.65.2.9 cv()

```
procedure, public multiparameter_base::meos::cv (
            class(meos) this,
            real, intent(in) t,
            real, intent(in) v )
```

Isochoric heat capacity.

Parameters

	this	Calling class
in	*v*	Temperature (K) and molar volume (m^3/mol)

Returns

[J/(mol∗K)]

6.65.2.10 densitysolver()

```
procedure, public multiparameter_base::meos::densitysolver (
            class(meos) this,
```

```
            real, intent(in) t_spec,
            real, intent(in) p_spec,
            integer, intent(in) phase_spec,
            real, intent(out) rho,
            integer, intent(out), optional phase_found,
            integer, intent(out), optional ierr )
```

Parameters

	this	The calling class.
in	*p_spec*	Temperature (K) and pressure (Pa)
in	*phase_spec*	Phase flag.
out	*rho*	Density (mol/m^3)

6.65.2.11 satdeltaestimate()

```
procedure(satdeltaestimate_intf), deferred, public multiparameter_base::meos::satdeltaestimate
(
            class(meos) this,
            real, intent(in) tau,
            integer, intent(in) phase )  [pure virtual]
```

An estimate delta_sat(tau_sat) for use in density solver.

Parameters

in	*tau*	Reduced temperature (-)
in	*phase*	Phase flag

Returns

Reduced density (-)

6.65.2.12 speed_of_sound()

```
procedure, public multiparameter_base::meos::speed_of_sound (
            class(meos) this,
            real, intent(in) t,
            real, intent(in) v )
```

[m/s]

Parameters

	this	Calling class
in	*v*	Temperature (K) and molar volume (m^3/mol)

Returns

[m/s]

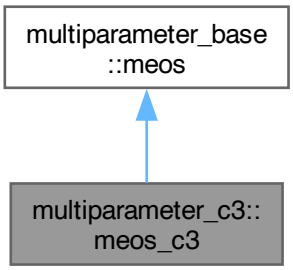
The documentation for this type was generated from the following file:

- multiparameter_base.f90

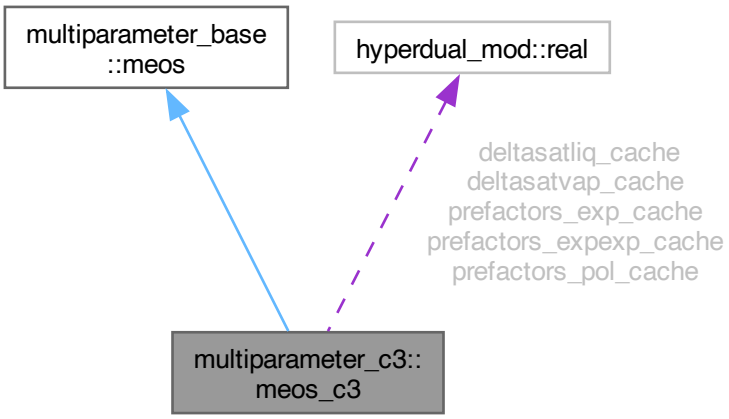
6.66 multiparameter_c3::meos_c3 Type Reference

C3 multiparameter equations of state (Lemmon, McLinden and Wagner 2009).

Inheritance diagram for multiparameter_c3::meos_c3:



Collaboration diagram for multiparameter_c3::meos_c3:



Public Member Functions

- procedure, public alpha0derivs_taudelta (this, delta, tau, alp0)
- procedure, public alpharesderivs_taudelta (this, delta, tau, alpr)
- procedure, public **satdeltaestimate** (this, tau, phase)
- procedure, public **init** (this, use_rgas_fit)
- procedure, public **alpha0_hd_taudelta** (this, delta, tau)
- procedure, public **alphares_hd_taudelta** (this, delta, tau)
- procedure, pass, public **assign_meos** (this, other)

Public Member Functions inherited from multiparameter_base::meos

- procedure, public **mp_pressure** (this, rho, t, p, p_rho, p_t)

  Pressure and (optionally) its derivatives.
- procedure, public **alpha_to_f_conversion** (this, t, v, n, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)

- procedure, public **calc_f** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- procedure, public **calc_fid** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)

  Calculate reduced ideal Helmholtz energy and differentials.
- procedure, public **calc_zfac** (this, t, p, n, phase, z, z_t, z_p, z_n)
- procedure, public **calc_lnphi** (this, t, p, n, phase, lnphi, lnphi_t, lnphi_p, lnphi_n)
- procedure, public **calc_entropy** (this, t, p, n, phase, s, s_t, s_p, s_n, residual)
- procedure, public **calc_enthalpy** (this, t, p, n, phase, h, h_t, h_p, h_n, residual)
- procedure, public **calc_resgibbs** (this, t, p, n, phase, g, g_t, g_p, g_n)
- procedure, public densitysolver (this, t_spec, p_spec, phase_spec, rho, phase_found, ierr)
- procedure, public alphaderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, residual)
- procedure, public alpharesderivs_tv (this, t, v, alpr, alpr_t, alpr_v, alpr_tt, alpr_tv, alpr_vv, alpr_n, alpr_tn, alpr_vn, alpr_nn)
- procedure, public alphaidderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, alp_n, alp_tn, alp_vn, alp_nn)
- procedure, public **getcritpoint** (this, tcrit, pcrit, rhocrit)
- procedure(init_intf), deferred, public **init** (this, use_rgas_fit)

  Initiate compName, critical point and triple point.
- procedure, public cv (this, t, v)

  Isochoric heat capacity.
- procedure, public cp (this, t, v)

  Isobaric heat capacity.
- procedure, public speed_of_sound (this, t, v)

  [m/s]
- procedure, public **get_ref_state_spec** (this, ref_state, t, p, phase, solve)

  Get specification for the current reference state.
- procedure, public **set_ref_state** (this, t, p, v, h, s)

  Set reference state.
- procedure(satdeltaestimate_intf), deferred, public satdeltaestimate (this, tau, phase)

  An estimate delta_sat(tau_sat) for use in density solver.
- procedure(alpha0derivs_intf), deferred, public alpha0derivs_taudelta (this, delta, tau, alp0)

  *[d^{j}alpha0/(d_tau)^j]*tau^j*
- procedure(alpharesderivs_intf), deferred, public alpharesderivs_taudelta (this, delta, tau, alpr)

  *[d^{i+j}alphaRes/(d_delta)^i(d_tau)^j]*delta^i*tau^j*
- procedure(alpha0_hd_intf), deferred, public alpha0_hd_taudelta (this, delta, tau)

  Calculate alpha0 using hyperdual numbers.
- procedure(alphares_hd_intf), deferred, public alphares_hd_taudelta (this, delta, tau)

  Calculate alphaRes using hyperdual numbers.
- procedure, public **assign_meos_base** (this, other)
- procedure(assign_meos_intf), deferred, pass, public **assign_meos** (this, other)
- generic, public **assignment** assign_meos

Public Attributes

- real **deltasatliq_cache**
- real **deltasatvap_cache**
- real, dimension(1:uppol) **prefactors_pol_cache**
- real, dimension(uppol+1:upexp) **prefactors_exp_cache**
- real, dimension(upexp+1:upexpexp) **prefactors_expexp_cache**

Public Attributes inherited from multiparameter_base::meos

- character(len=20), public **compname**

  Parameters in SI units. These are set in the deferred init routine.
- real, public **tc**
- real, public **pc**
- real, public **rc**
- real, public **acf**
- real, public **t_triple**
- real, public **p_triple**
- real, public **rholiq_triple**
- real, public **rhovap_triple**
- real, public **molarmass**

  (kg/mol)
- real, public **maxt**
- real, public **maxp**

  (K), (Pa)
- real, public **rgas_meos** = Rgas_default
- real, public **rgas_fit**

6.66.1 Detailed Description

C3 multiparameter equations of state (Lemmon, McLinden and Wagner 2009).

6.66.2 Member Function/Subroutine Documentation

6.66.2.1 alpha0derivs_taudelta()

```
procedure, public multiparameter_c3::meos_c3::alpha0derivs_taudelta (
              class(meos_c3) this,
              real, intent(in) delta,
              real, intent(in) tau,
              real, dimension(0:2,0:2), intent(out) alp0 )
```

Parameters

out	*alp0*	alp0(i,j) = [(d_delta)^i(d_tau)^j alpha0]*delta^i*tau^j

6.66.2.2 alpharesderivs_taudelta()

```
procedure, public multiparameter_c3::meos_c3::alpharesderivs_taudelta (
              class(meos_c3) this,
              real, intent(in) delta,
              real, intent(in) tau,
              real, dimension(0:2,0:2), intent(out) alpr )
```

Parameters

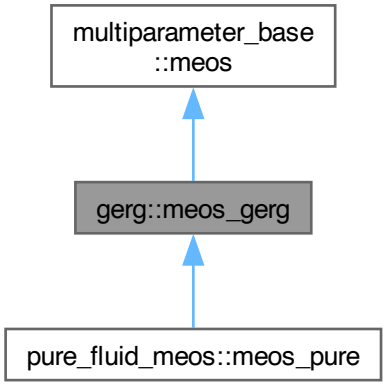
out	*alpr*	alpr(i,j) = (d_delta)^i(d_tau)^j alphaRes

The documentation for this type was generated from the following file:

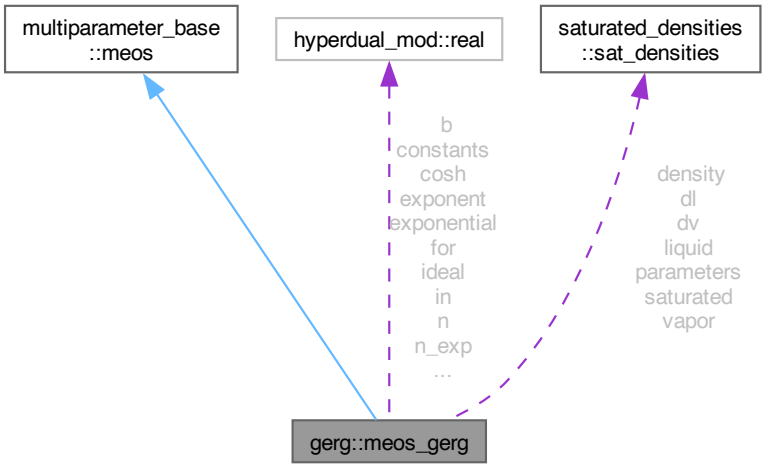
- multiparameter_c3.f90

6.67 gerg::meos_gerg Type Reference

GERG-2008 multiparameter equations of state.
Inheritance diagram for gerg::meos_gerg:



Collaboration diagram for gerg::meos_gerg:



Public Member Functions

- procedure, public alpha0derivs_taudelta (this, delta, tau, alp0)

  Specific reduced Helmholtz energy - ideal gas contributuion.
- procedure, public alpharesderivs_taudelta (this, delta, tau, alpr)

  Specific residual reduced Helmholtz energy.
- procedure, public **satdeltaestimate** (this, tau, phase)

  Estimate saturated densities for density solver.

- procedure, public **init** (this, use_rgas_fit)
- procedure, public **allocate_param** (this)
- procedure, public **alpha0_hd_taudelta** (this, delta, tau)

  Specific reduced Helmholtz energy - ideal gas contributuion. Hyperdual numbers.
- procedure, public **alphares_hd_taudelta** (this, delta, tau)

  Specific residual reduced Helmholtz energy. Hyperdual numbers.
- procedure, pass, public **assign_meos** (this, other)

Public Member Functions inherited from multiparameter_base::meos

- procedure, public **mp_pressure** (this, rho, t, p, p_rho, p_t)

  Pressure and (optionally) its derivatives.
- procedure, public **alpha_to_f_conversion** (this, t, v, n, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- procedure, public **calc_f** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- procedure, public **calc_fid** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)

  Calculate reduced ideal Helmholtz energy and differentials.
- procedure, public **calc_zfac** (this, t, p, n, phase, z, z_t, z_p, z_n)
- procedure, public **calc_lnphi** (this, t, p, n, phase, lnphi, lnphi_t, lnphi_p, lnphi_n)
- procedure, public **calc_entropy** (this, t, p, n, phase, s, s_t, s_p, s_n, residual)
- procedure, public **calc_enthalpy** (this, t, p, n, phase, h, h_t, h_p, h_n, residual)
- procedure, public **calc_resgibbs** (this, t, p, n, phase, g, g_t, g_p, g_n)
- procedure, public densitysolver (this, t_spec, p_spec, phase_spec, rho, phase_found, ierr)
- procedure, public alphaderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, residual)
- procedure, public alpharesderivs_tv (this, t, v, alpr, alpr_t, alpr_v, alpr_tt, alpr_tv, alpr_vv, alpr_n, alpr_tn, alpr_vn, alpr_nn)
- procedure, public alphaidderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, alp_n, alp_tn, alp_vn, alp_nn)
- procedure, public **getcritpoint** (this, tcrit, pcrit, rhocrit)
- procedure(init_intf), deferred, public **init** (this, use_rgas_fit)

  Initiate compName, critical point and triple point.
- procedure, public cv (this, t, v)

  Isochoric heat capacity.
- procedure, public cp (this, t, v)

  Isobaric heat capacity.
- procedure, public speed_of_sound (this, t, v)

  [m/s]
- procedure, public **get_ref_state_spec** (this, ref_state, t, p, phase, solve)

  Get specification for the current reference state.
- procedure, public **set_ref_state** (this, t, p, v, h, s)

  Set reference state.
- procedure(satdeltaestimate_intf), deferred, public satdeltaestimate (this, tau, phase)

  An estimate delta_sat(tau_sat) for use in density solver.
- procedure(alpha0derivs_intf), deferred, public alpha0derivs_taudelta (this, delta, tau, alp0)

  *[d^{j}alpha0/(d_tau)^j]*tau^j*
- procedure(alpharesderivs_intf), deferred, public alpharesderivs_taudelta (this, delta, tau, alpr)

  *[d^{i+j}alphaRes/(d_delta)^i(d_tau)^j]*delta^i*tau^j*
- procedure(alpha0_hd_intf), deferred, public alpha0_hd_taudelta (this, delta, tau)

  Calculate alpha0 using hyperdual numbers.
- procedure(alphares_hd_intf), deferred, public alphares_hd_taudelta (this, delta, tau)

  Calculate alphaRes using hyperdual numbers.
- procedure, public **assign_meos_base** (this, other)
- procedure(assign_meos_intf), deferred, pass, public **assign_meos** (this, other)
- generic, public **assignment** assign_meos

Public Attributes

- integer **i_comp** = 0
- integer **index**
- integer, dimension(:), allocatable **of**
- integer **component**
- integer, dimension(:), allocatable **in**
- integer **gergdatadb**
- integer **n_cosh** = 0
- integer **number**
- integer, dimension(:), allocatable **cosh**
- integer, dimension(:), allocatable **terms**
- integer **n_sinh** = 0
- integer **and**
- integer, dimension(:), allocatable **sinh**
- real, dimension(3) **n**
- real **constants**
- real, dimension(:), allocatable **v**
- real, dimension(:), allocatable **prefactor**
- real, dimension(:), allocatable **ideal**
- real, dimension(:), allocatable **terms**
- real, dimension(:), allocatable **b**
- real, dimension(:), allocatable **cosh**
- real, dimension(:), allocatable **sinh**
- real, dimension(:), allocatable **parameter**
- real, dimension(:), allocatable **for**
- integer **uppol** = 0
- integer, dimension(:), allocatable **polynomial**
- integer **upexp** = 0
- integer, dimension(:), allocatable **exponential**
- real, dimension(:), allocatable **n_pol**
- real, dimension(:), allocatable **polynomial**
- real, dimension(:), allocatable **n_exp**
- real, dimension(:), allocatable **exponential**
- real, dimension(:), allocatable **t_pol**
- real, dimension(:), allocatable **tau**
- real, dimension(:), allocatable **exponent**
- real, dimension(:), allocatable **t_exp**
- real, dimension(:), allocatable **in**
- integer, dimension(:), allocatable **d_pol**
- integer, dimension(:), allocatable **delta**
- integer, dimension(:), allocatable **exponent**
- integer, dimension(:), allocatable **d_exp**
- integer, dimension(:), allocatable **for**
- integer, dimension(:), allocatable **prefactor**
- integer, dimension(:), allocatable **l_exp**
- integer, dimension(:), allocatable **term**
- type(sat_densities) **dl**
- type(sat_densities) **parameters**
- type(sat_densities) **liquid**
- type(sat_densities) **saturated**
- type(sat_densities) **density**
- type(sat_densities) **dv**
- type(sat_densities) **vapor**

Public Attributes inherited from multiparameter_base::meos

- character(len=20), public **compname**

  Parameters in SI units. These are set in the deferred init routine.
- real, public **tc**
- real, public **pc**
- real, public **rc**
- real, public **acf**
- real, public **t_triple**
- real, public **p_triple**
- real, public **rholiq_triple**
- real, public **rhovap_triple**
- real, public **molarmass**

  (kg/mol)
- real, public **maxt**
- real, public **maxp**

  (K), (Pa)
- real, public **rgas_meos** = Rgas_default
- real, public **rgas_fit**

6.67.1 Detailed Description

GERG-2008 multiparameter equations of state.

6.67.2 Member Function/Subroutine Documentation

6.67.2.1 alpha0derivs_taudelta()

```
procedure, public gerg::meos_gerg::alpha0derivs_taudelta (
             class(meos_gerg) this,
             real, intent(in) delta,
             real, intent(in) tau,
             real, dimension(0:2,0:2), intent(out) alp0 )
```

Specific reduced Helmholtz energy - ideal gas contributuion.

Parameters

out	*alp0*	alp0(i,j) = [(d_delta)^i(d_tau)^j alpha0]*delta^i*tau^j

6.67.2.2 alpharesderivs_taudelta()

```
procedure, public gerg::meos_gerg::alpharesderivs_taudelta (
             class(meos_gerg) this,
             real, intent(in) delta,
             real, intent(in) tau,
             real, dimension(0:2,0:2), intent(out) alpr )
```

Specific residual reduced Helmholtz energy.

Parameters

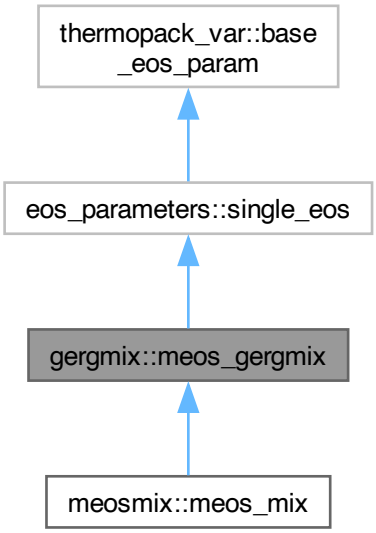
out	*alpr*	alpr(i,j) = (d_delta)^i(d_tau)^j alphaRes

The documentation for this type was generated from the following file:

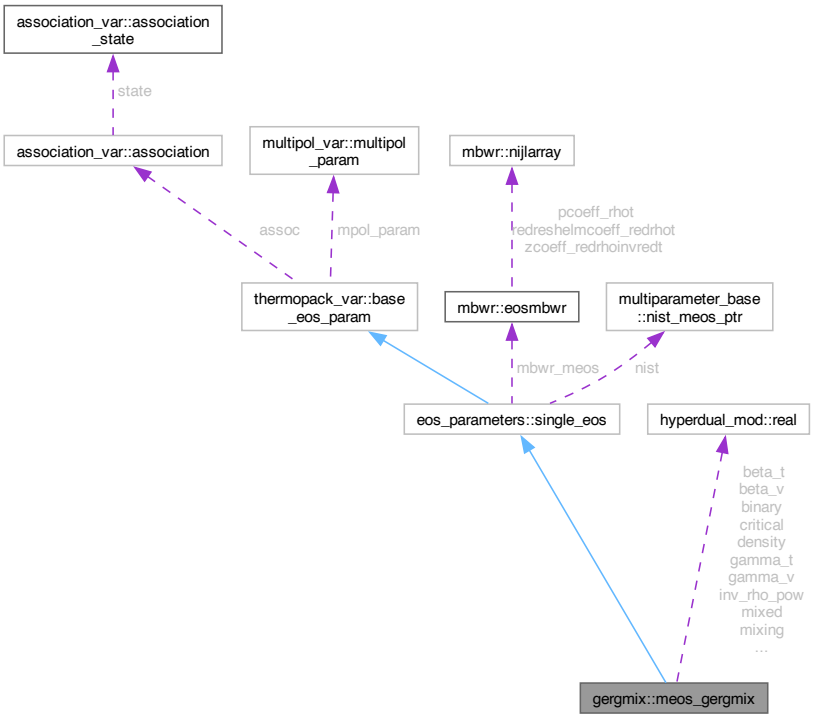
- gerg.f90

6.68 gergmix::meos_gergmix Type Reference

GERG-2008 multiparameter equations of state.
Inheritance diagram for gergmix::meos_gergmix:



Collaboration diagram for gergmix::meos_gergmix:



Public Member Functions

- procedure, public **allocate_param** (this, nc)
- procedure, public **alpha0_hd** (this, x, rho, t)

  Specific reduced Helmholtz energy - ideal gas contributuion.
- procedure, public **alphares_hd** (this, x, delta, tau)

  Specific residual reduced Helmholtz energy.
- procedure, public **zfac** (eos, t, p, z, phase, zfac, phase_found)
- procedure, pass, public **assign_meos** (this, other)
- procedure, public **calc_delta** (this, x, rho)

  Calculate mixture delta.
- procedure, public **calc_tau** (this, x, t)

  Calculate mixture tau.
- procedure, public **calc_del_alpha_r** (this, x, tau, delta)

  Calculate departure function for mixing.
- procedure, public **pressure** (this, rho, x, t_spec, p, p_rho, p_rhorho)

  Pressure method using hyperdual numbers for differentials.
- procedure, public densitysolver (this, x, t_spec, p_spec, phase_spec, rho, phase_found, ierr)

  Density solver. Specified T,P and composition,.
- procedure, public fake_density (this, x, t_spec, p_spec, phase_spec, rho, ierr, phase_found)

  Calculate fake phase.

Public Member Functions inherited from eos_parameters::single_eos

- procedure, public **dealloc** (eos)
- procedure, public allocate_and_init (eos, nc, eos_label)
- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from thermopack_var::base_eos_param

- procedure(allocate_and_init_intf), deferred, public allocate_and_init (eos, nc, eos_label)
- procedure, public **dealloc** (eos)
- procedure(assign_intf), deferred, pass, public **assign_eos** (this, other)
- generic, public **assignment** assign_eos
- procedure, public **assign_base_eos_param** (this, other)

Public Attributes

- real, dimension(:, :), allocatable **inv_rho_pow**
- real, dimension(:, :), allocatable **binary**
- real, dimension(:, :), allocatable **mixed**
- real, dimension(:, :), allocatable **critical**
- real, dimension(:, :), allocatable **density**
- real, dimension(:, :), allocatable **tc_prod_sqrt**
- real, dimension(:, :), allocatable **temperature**
- real, dimension(:, :), allocatable **beta_t**
- real, dimension(:, :), allocatable **mixing**
- real, dimension(:, :), allocatable **parameter**
- real, dimension(:, :), allocatable **beta_v**
- real, dimension(:, :), allocatable **gamma_t**
- real, dimension(:, :), allocatable **gamma_v**
- integer, dimension(:, :), allocatable **mix_data_index**
- integer, dimension(:, :), allocatable **index**
- integer, dimension(:, :), allocatable **in**
- integer, dimension(:, :), allocatable **database**

Public Attributes inherited from eos_parameters::single_eos

- type(eosmbwr), dimension(:), allocatable **mbwr_meos**
- type(nist_meos_ptr), dimension(:), allocatable **nist**

Public Attributes inherited from thermopack_var::base_eos_param

- character(len=eosid_len) **eosid**

  Eos identifier.
- integer **eosidx**

  Eos group index.
- integer **subeosidx**

  Eos sub-index.
- integer **volumeshiftid** = 0

  0: No volume shift, 1:Peneloux shift
- logical **iselectrolyteeos** = .false.

  Used to enable electrolytes.
- type(association), pointer **assoc** => NULL()
- type(multipol_param), pointer **mpol_param** => NULL()

6.68.1 Detailed Description

GERG-2008 multiparameter equations of state.

6.68.2 Member Function/Subroutine Documentation

6.68.2.1 densitysolver()

```
procedure, public gergmix::meos_gergmix::densitysolver (
             class(meos_gergmix) this,
             real, dimension(nce), intent(in) x,
             real, intent(in) t_spec,
             real, intent(in) p_spec,
             integer, intent(in) phase_spec,
             real, intent(out) rho,
             integer, intent(out), optional phase_found,
             integer, intent(out), optional ierr )
```

Density solver. Specified T,P and composition,.

Parameters

	this	The calling class.
in	*x*	Temperature (K) and pressure (Pa)
in	*phase_spec*	Phase flag.
out	*rho*	Density (mol/m^3)

6.68.2.2 fake_density()

```
procedure, public gergmix::meos_gergmix::fake_density (
             class(meos_gergmix) this,
             real, dimension(nce), intent(in) x,
             real, intent(in) t_spec,
             real, intent(in) p_spec,
             integer, intent(in) phase_spec,
             real, intent(out) rho,
```

```
integer, intent(out) ierr,
integer, intent(out), optional phase_found )
```

Calculate fake phase.

Parameters

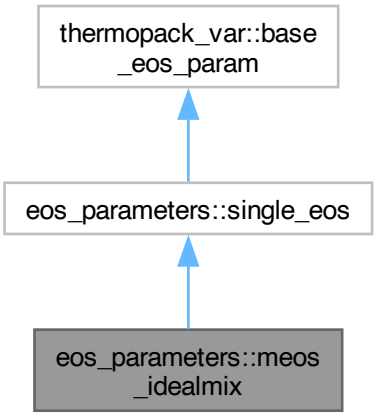
	this	The calling class.
in	*x*	Temperature (K) and pressure (Pa)
in	*phase_spec*	Phase flag
out	*rho*	Density (mol/m^3)

The documentation for this type was generated from the following file:

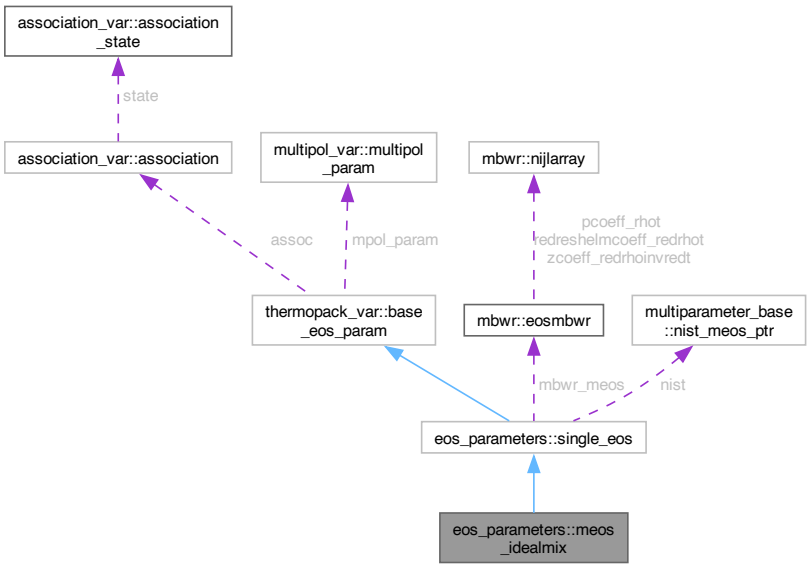
- gergmix.f90

6.69 eos_parameters::meos_idealmix Type Reference

Inheritance diagram for eos_parameters::meos_idealmix:



Collaboration diagram for eos_parameters::meos_idealmix:



Additional Inherited Members

Public Member Functions inherited from eos_parameters::single_eos

- procedure, public **dealloc** (eos)
- procedure, public allocate_and_init (eos, nc, eos_label)
- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from thermopack_var::base_eos_param

- procedure(allocate_and_init_intf), deferred, public allocate_and_init (eos, nc, eos_label)
- procedure, public **dealloc** (eos)
- procedure(assign_intf), deferred, pass, public **assign_eos** (this, other)
- generic, public **assignment** assign_eos
- procedure, public **assign_base_eos_param** (this, other)

Public Attributes inherited from eos_parameters::single_eos

- type(eosmbwr), dimension(:), allocatable **mbwr_meos**
- type(nist_meos_ptr), dimension(:), allocatable **nist**

Public Attributes inherited from thermopack_var::base_eos_param

- character(len=eosid_len) **eosid**

  Eos identifier.
- integer **eosidx**

  Eos group index.
- integer **subeosidx**

  Eos sub-index.
- integer **volumeshiftid** = 0

  0: No volume shift, 1:Peneloux shift

- logical **iselectrolyteeos** = .false.

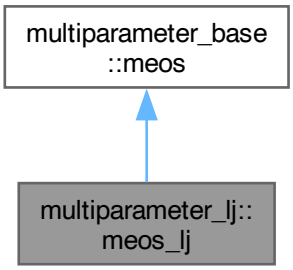
  Used to enable electrolytes.
- type(association), pointer **assoc** => NULL()
- type(multipol_param), pointer **mpol_param** => NULL()

The documentation for this type was generated from the following file:

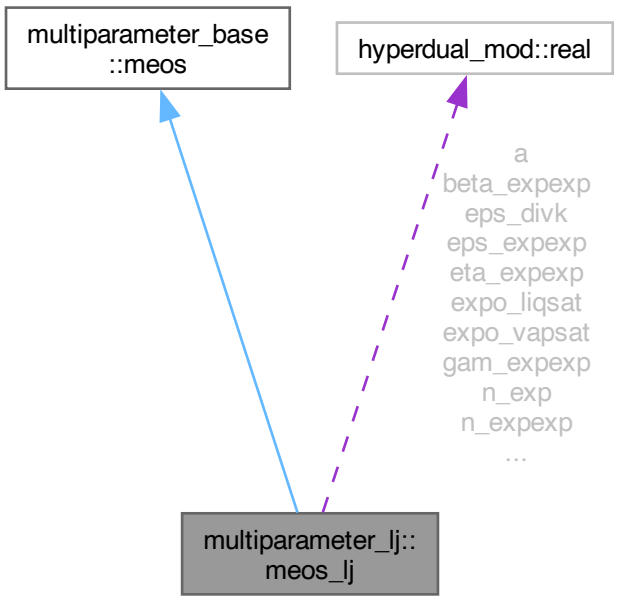
- eos_parameters.f90

6.70 multiparameter_lj::meos_lj Type Reference

LJ multiparameter equations of state (Thol, Vrabec and Span).
Inheritance diagram for multiparameter_lj::meos_lj:



Collaboration diagram for multiparameter_lj::meos_lj:



Public Member Functions

- procedure, public alpha0derivs_taudelta (this, delta, tau, alp0)
- procedure, public alpharesderivs_taudelta (this, delta, tau, alpr)
- procedure, public **satdeltaestimate** (this, tau, phase)
- procedure, public **init** (this, use_rgas_fit)
- procedure, public **alpha0_hd_taudelta** (this, delta, tau)
- procedure, public **alphares_hd_taudelta** (this, delta, tau)
- procedure, pass, public **assign_meos** (this, other)

Public Member Functions inherited from multiparameter_base::meos

- procedure, public **mp_pressure** (this, rho, t, p, p_rho, p_t)

  Pressure and (optionally) its derivatives.
- procedure, public **alpha_to_f_conversion** (this, t, v, n, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- procedure, public **calc_f** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- procedure, public **calc_fid** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)

  Calculate reduced ideal Helmholtz energy and differentials.
- procedure, public **calc_zfac** (this, t, p, n, phase, z, z_t, z_p, z_n)
- procedure, public **calc_lnphi** (this, t, p, n, phase, lnphi, lnphi_t, lnphi_p, lnphi_n)
- procedure, public **calc_entropy** (this, t, p, n, phase, s, s_t, s_p, s_n, residual)
- procedure, public **calc_enthalpy** (this, t, p, n, phase, h, h_t, h_p, h_n, residual)
- procedure, public **calc_resgibbs** (this, t, p, n, phase, g, g_t, g_p, g_n)
- procedure, public densitysolver (this, t_spec, p_spec, phase_spec, rho, phase_found, ierr)
- procedure, public alphaderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, residual)
- procedure, public alpharesderivs_tv (this, t, v, alpr, alpr_t, alpr_v, alpr_tt, alpr_tv, alpr_vv, alpr_n, alpr_tn, alpr_vn, alpr_nn)
- procedure, public alphaidderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, alp_n, alp_tn, alp_vn, alp_nn)
- procedure, public **getcritpoint** (this, tcrit, pcrit, rhocrit)
- procedure(init_intf), deferred, public **init** (this, use_rgas_fit)

  Initiate compName, critical point and triple point.
- procedure, public cv (this, t, v)

  Isochoric heat capacity.
- procedure, public cp (this, t, v)

  Isobaric heat capacity.
- procedure, public speed_of_sound (this, t, v)

  [m/s]
- procedure, public **get_ref_state_spec** (this, ref_state, t, p, phase, solve)

  Get specification for the current reference state.
- procedure, public **set_ref_state** (this, t, p, v, h, s)

  Set reference state.
- procedure(satdeltaestimate_intf), deferred, public satdeltaestimate (this, tau, phase)

  An estimate delta_sat(tau_sat) for use in density solver.
- procedure(alpha0derivs_intf), deferred, public alpha0derivs_taudelta (this, delta, tau, alp0)

  *[d^{j}alpha0/(d_tau)^j]*tau^j*
- procedure(alpharesderivs_intf), deferred, public alpharesderivs_taudelta (this, delta, tau, alpr)

  *[d^{i+j}alphaRes/(d_delta)^i(d_tau)^j]*delta^i*tau^j*
- procedure(alpha0_hd_intf), deferred, public alpha0_hd_taudelta (this, delta, tau)

  Calculate alpha0 using hyperdual numbers.
- procedure(alphares_hd_intf), deferred, public alphares_hd_taudelta (this, delta, tau)

  Calculate alphaRes using hyperdual numbers.
- procedure, public **assign_meos_base** (this, other)
- procedure(assign_meos_intf), deferred, pass, public **assign_meos** (this, other)
- generic, public **assignment** assign_meos

Public Attributes

- real **sigma**
- real **eps_divk**
- real, dimension(:), allocatable **a**
- integer **uppol** = 0
- integer **upexp** = 0
- integer **upexpexp** = 0
- real, dimension(:), allocatable **n_pol**
- real, dimension(:), allocatable **n_exp**
- real, dimension(:), allocatable **n_expexp**
- real, dimension(:), allocatable **t_pol**
- real, dimension(:), allocatable **t_exp**
- real, dimension(:), allocatable **t_expexp**
- integer, dimension(:), allocatable **d_pol**
- integer, dimension(:), allocatable **d_exp**
- integer, dimension(:), allocatable **d_expexp**
- integer, dimension(:), allocatable **l_exp**
- real, dimension(:), allocatable **eta_expexp**
- real, dimension(:), allocatable **beta_expexp**
- real, dimension(:), allocatable **gam_expexp**
- real, dimension(:), allocatable **eps_expexp**
- integer **satp_liq** = 0
- integer **satp_vap** = 0
- real, dimension(:), allocatable **n_liqsat**
- real, dimension(:), allocatable **expo_liqsat**
- real, dimension(:), allocatable **n_vapsat**
- real, dimension(:), allocatable **expo_vapsat**

Public Attributes inherited from multiparameter_base::meos

- character(len=20), public **compname**

  Parameters in SI units. These are set in the deferred init routine.
- real, public **tc**
- real, public **pc**
- real, public **rc**
- real, public **acf**
- real, public **t_triple**
- real, public **p_triple**
- real, public **rholiq_triple**
- real, public **rhovap_triple**
- real, public **molarmass**

  (kg/mol)
- real, public **maxt**
- real, public **maxp**

  (K), (Pa)
- real, public **rgas_meos** = Rgas_default
- real, public **rgas_fit**

6.70.1 Detailed Description

LJ multiparameter equations of state (Thol, Vrabec and Span).

6.70.2 Member Function/Subroutine Documentation

6.70.2.1 alpha0derivs_taudelta()

```
procedure, public multiparameter_lj::meos_lj::alpha0derivs_taudelta (
             class(meos_lj) this,
             real, intent(in) delta,
             real, intent(in) tau,
             real, dimension(0:2,0:2), intent(out) alp0 )
```

Parameters

out	*alp0*	alp0(i,j) = [(d_delta)^i(d_tau)^j alpha0]∗delta^i∗tau^j

6.70.2.2 alpharesderivs_taudelta()

```
procedure, public multiparameter_lj::meos_lj::alpharesderivs_taudelta (
             class(meos_lj) this,
             real, intent(in) delta,
             real, intent(in) tau,
             real, dimension(0:2,0:2), intent(out) alpr )
```

Parameters

out	*alpr*	alpr(i,j) = (d_delta)^i(d_tau)^j alphaRes

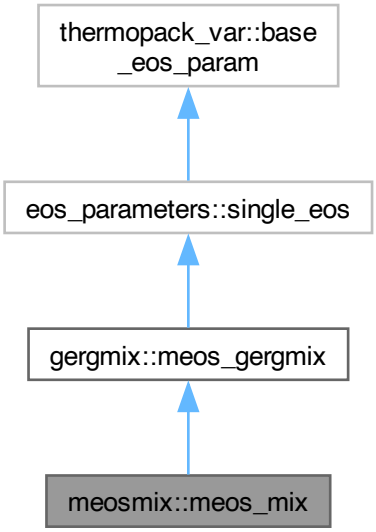
The documentation for this type was generated from the following file:

- multiparameter_lj.f90

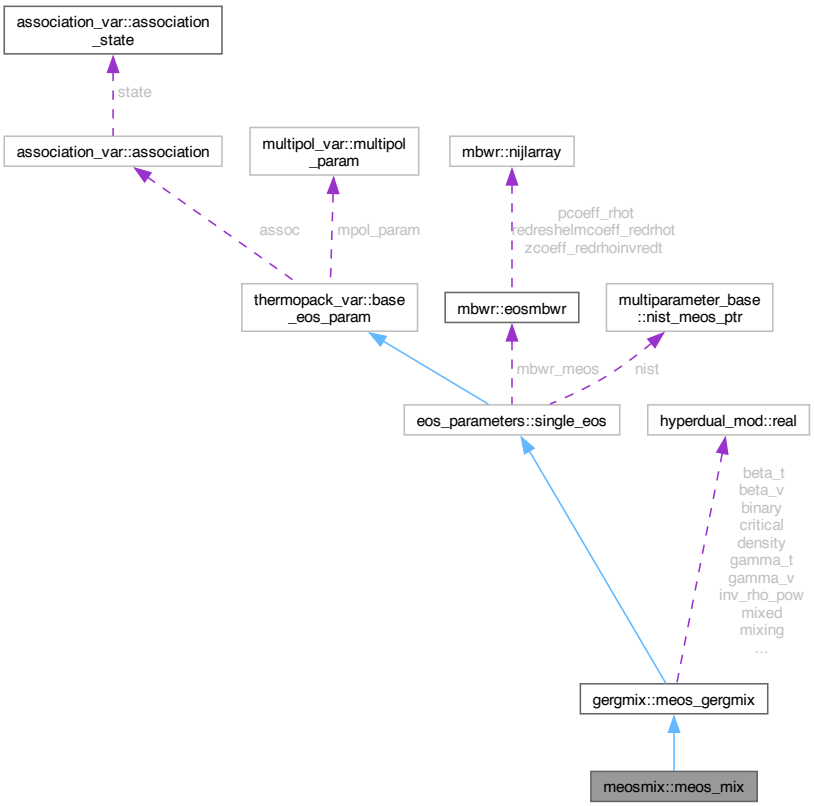
6.71 meosmix::meos_mix Type Reference

Multiparameter equations of state.

Inheritance diagram for meosmix::meos_mix:



Collaboration diagram for meosmix::meos_mix:



Public Member Functions

- procedure, public **rgas_mix** (this, x)
- procedure, public **calc_del_alpha_r** (this, x, tau, delta)
- procedure, public fake_density (this, x, t_spec, p_spec, phase_spec, rho, ierr, phase_found)

  Calculate extremas in pressure and determine if a fake root is needed.

Public Member Functions inherited from gergmix::meos_gergmix

- procedure, public **allocate_param** (this, nc)
- procedure, public **alpha0_hd** (this, x, rho, t)

  Specific reduced Helmholtz energy - ideal gas contributuion.
- procedure, public **alphares_hd** (this, x, delta, tau)

  Specific residual reduced Helmholtz energy.
- procedure, public **zfac** (eos, t, p, z, phase, zfac, phase_found)
- procedure, pass, public **assign_meos** (this, other)
- procedure, public **calc_delta** (this, x, rho)

  Calculate mixture delta.
- procedure, public **calc_tau** (this, x, t)

  Calculate mixture tau.
- procedure, public **calc_del_alpha_r** (this, x, tau, delta)

  Calculate departure function for mixing.
- procedure, public **pressure** (this, rho, x, t_spec, p, p_rho, p_rhorho)

  Pressure method using hyperdual numbers for differentials.
- procedure, public densitysolver (this, x, t_spec, p_spec, phase_spec, rho, phase_found, ierr)

  Density solver. Specified T,P and composition,.
- procedure, public fake_density (this, x, t_spec, p_spec, phase_spec, rho, ierr, phase_found)

  Calculate fake phase.

Public Member Functions inherited from eos_parameters::single_eos

- procedure, public **dealloc** (eos)
- procedure, public allocate_and_init (eos, nc, eos_label)
- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from thermopack_var::base_eos_param

- procedure(allocate_and_init_intf), deferred, public allocate_and_init (eos, nc, eos_label)
- procedure, public **dealloc** (eos)
- procedure(assign_intf), deferred, pass, public **assign_eos** (this, other)
- generic, public **assignment** assign_eos
- procedure, public **assign_base_eos_param** (this, other)

Additional Inherited Members

Public Attributes inherited from gergmix::meos_gergmix

- real, dimension(:, :), allocatable **inv_rho_pow**
- real, dimension(:, :), allocatable **binary**
- real, dimension(:, :), allocatable **mixed**
- real, dimension(:, :), allocatable **critical**
- real, dimension(:, :), allocatable **density**
- real, dimension(:, :), allocatable **tc_prod_sqrt**
- real, dimension(:, :), allocatable **temperature**
- real, dimension(:, :), allocatable **beta_t**

- real, dimension(:, :), allocatable **mixing**
- real, dimension(:, :), allocatable **parameter**
- real, dimension(:, :), allocatable **beta_v**
- real, dimension(:, :), allocatable **gamma_t**
- real, dimension(:, :), allocatable **gamma_v**
- integer, dimension(:, :), allocatable **mix_data_index**
- integer, dimension(:, :), allocatable **index**
- integer, dimension(:, :), allocatable **in**
- integer, dimension(:, :), allocatable **database**

Public Attributes inherited from eos_parameters::single_eos

- type(eosmbwr), dimension(:), allocatable **mbwr_meos**
- type(nist_meos_ptr), dimension(:), allocatable **nist**

Public Attributes inherited from thermopack_var::base_eos_param

- character(len=eosid_len) **eosid**

  Eos identifier.
- integer **eosidx**

  Eos group index.
- integer **subeosidx**

  Eos sub-index.
- integer **volumeshiftid** = 0

  0: No volume shift, 1:Peneloux shift
- logical **iselectrolyteeos** = .false.

  Used to enable electrolytes.
- type(association), pointer **assoc** => NULL()
- type(multipol_param), pointer **mpol_param** => NULL()

6.71.1 Detailed Description

Multiparameter equations of state.

6.71.2 Member Function/Subroutine Documentation

6.71.2.1 fake_density()

```
procedure, public meosmix::meos_mix::fake_density (
             class(meos_mix) this,
             real, dimension(nce), intent(in) x,
             real, intent(in) t_spec,
             real, intent(in) p_spec,
             integer, intent(in) phase_spec,
             real, intent(out) rho,
             integer, intent(out) ierr,
             integer, intent(out), optional phase_found )
```

Calculate extremas in pressure and determine if a fake root is needed.

Parameters

	this	The calling class.
in	*x*	Temperature (K) and pressure (Pa)
in	*phase_spec*	Phase flag
out	*rho*	Density (mol/m^3)
out	*phase_found*	Phase flag of detected phase

The documentation for this type was generated from the following file:

- meosmix.f90

6.72 meosmixdb::meos_mix_data Type Reference

Public Attributes

- character(len=uid_len) **ident1**

  The component ID.
- character(len=uid_len) **ident2**

  The component ID.
- character(len=bibref_len) **bibref**

  Digital Object Identifier (DOI) or reference.
- real **fij**

  Departure function parameter.
- integer **num_mix**

  Number of parameters.
- real, dimension(12) **n_mix**
- real, dimension(12) **t_mix**
- integer, dimension(12) **d_mix**
- integer, dimension(12) **l_mix**
- real, dimension(12) **eta_mix**
- real, dimension(12) **gamma_mix**
- real, dimension(12) **epsilon_mix**
- real, dimension(12) **beta_mix**
- integer **num_exp**

  Number of exponential terms.
- integer **num_gauss**

  Number of Gaussian terms.

The documentation for this type was generated from the following file:

- meosmixdb.f90

6.73 meosmixdb::meos_mix_reducing Type Reference

Public Attributes

- character(len=uid_len) **ident1**

  The component ID.
- character(len=uid_len) **ident2**

  The component ID.
- character(len=bibref_len) **bibref**

  Digital Object Identifier (DOI) or reference.
- real **beta_v**

  Reducing density parameter.
- real **gamma_v**

  Reducing density parameter.
- real **beta_t**

  Reducing temperature parameter.
- real **gamma_t**

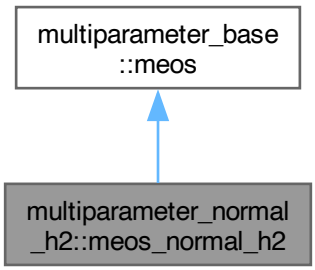
  Reducing temperature parameter.

The documentation for this type was generated from the following file:

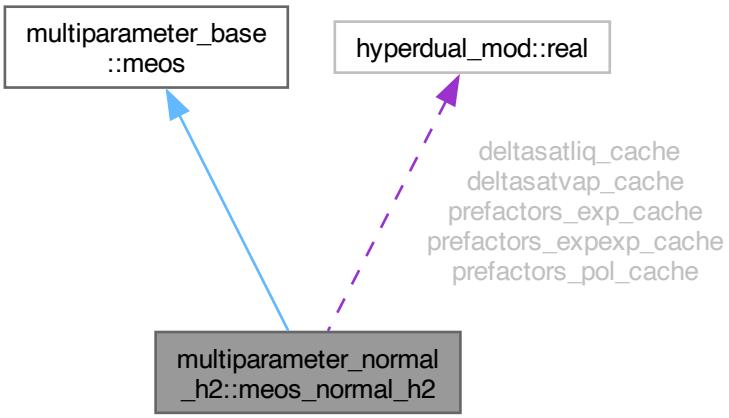
- meosmixdb.f90

6.74 multiparameter_normal_h2::meos_normal_h2 Type Reference

Inheritance diagram for multiparameter_normal_h2::meos_normal_h2:



Collaboration diagram for multiparameter_normal_h2::meos_normal_h2:



Public Member Functions

- procedure, public alpha0derivs_taudelta (this, delta, tau, alp0)
- procedure, public alpharesderivs_taudelta (this, delta, tau, alpr)
- procedure, public **satdeltaestimate** (this, tau, phase)
- procedure, public **init** (this, use_rgas_fit)
- procedure, public **alpha0_hd_taudelta** (this, delta, tau)
- procedure, public **alphares_hd_taudelta** (this, delta, tau)
- procedure, pass, public **assign_meos** (this, other)

Public Member Functions inherited from multiparameter_base::meos

- procedure, public **mp_pressure** (this, rho, t, p, p_rho, p_t)

  Pressure and (optionally) its derivatives.

- procedure, public **alpha_to_f_conversion** (this, t, v, n, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- procedure, public **calc_f** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- procedure, public **calc_fid** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)

  Calculate reduced ideal Helmholtz energy and differentials.
- procedure, public **calc_zfac** (this, t, p, n, phase, z, z_t, z_p, z_n)
- procedure, public **calc_lnphi** (this, t, p, n, phase, lnphi, lnphi_t, lnphi_p, lnphi_n)
- procedure, public **calc_entropy** (this, t, p, n, phase, s, s_t, s_p, s_n, residual)
- procedure, public **calc_enthalpy** (this, t, p, n, phase, h, h_t, h_p, h_n, residual)
- procedure, public **calc_resgibbs** (this, t, p, n, phase, g, g_t, g_p, g_n)
- procedure, public densitysolver (this, t_spec, p_spec, phase_spec, rho, phase_found, ierr)
- procedure, public alphaderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, residual)
- procedure, public alpharesderivs_tv (this, t, v, alpr, alpr_t, alpr_v, alpr_tt, alpr_tv, alpr_vv, alpr_n, alpr_tn, alpr_vn, alpr_nn)
- procedure, public alphaidderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, alp_n, alp_tn, alp_vn, alp_nn)
- procedure, public **getcritpoint** (this, tcrit, pcrit, rhocrit)
- procedure(init_intf), deferred, public **init** (this, use_rgas_fit)

  Initiate compName, critical point and triple point.
- procedure, public cv (this, t, v)

  Isochoric heat capacity.
- procedure, public cp (this, t, v)

  Isobaric heat capacity.
- procedure, public speed_of_sound (this, t, v)

  [m/s]
- procedure, public **get_ref_state_spec** (this, ref_state, t, p, phase, solve)

  Get specification for the current reference state.
- procedure, public **set_ref_state** (this, t, p, v, h, s)

  Set reference state.
- procedure(satdeltaestimate_intf), deferred, public satdeltaestimate (this, tau, phase)

  An estimate delta_sat(tau_sat) for use in density solver.
- procedure(alpha0derivs_intf), deferred, public alpha0derivs_taudelta (this, delta, tau, alp0)

  *[d^{j}alpha0/(d_tau)^j]*tau^j*
- procedure(alpharesderivs_intf), deferred, public alpharesderivs_taudelta (this, delta, tau, alpr)

  *[d^{i+j}alphaRes/(d_delta)^i(d_tau)^j]*delta^i*tau^j*
- procedure(alpha0_hd_intf), deferred, public alpha0_hd_taudelta (this, delta, tau)

  Calculate alpha0 using hyperdual numbers.
- procedure(alphares_hd_intf), deferred, public alphares_hd_taudelta (this, delta, tau)

  Calculate alphaRes using hyperdual numbers.
- procedure, public **assign_meos_base** (this, other)
- procedure(assign_meos_intf), deferred, pass, public **assign_meos** (this, other)
- generic, public **assignment** assign_meos

Public Attributes

- real **deltasatliq_cache**
- real **deltasatvap_cache**
- real, dimension(1:uppol) **prefactors_pol_cache**
- real, dimension(uppol+1:upexp) **prefactors_exp_cache**
- real, dimension(upexp+1:upexpexp) **prefactors_expexp_cache**

Public Attributes inherited from multiparameter_base::meos

- character(len=20), public **compname**

  Parameters in SI units. These are set in the deferred init routine.
- real, public **tc**
- real, public **pc**
- real, public **rc**
- real, public **acf**
- real, public **t_triple**
- real, public **p_triple**
- real, public **rholiq_triple**
- real, public **rhovap_triple**
- real, public **molarmass**

  (kg/mol)
- real, public **maxt**
- real, public **maxp**

  (K), (Pa)
- real, public **rgas_meos** = Rgas_default
- real, public **rgas_fit**

6.74.1 Member Function/Subroutine Documentation

6.74.1.1 alpha0derivs_taudelta()

```
procedure, public multiparameter_normal_h2::meos_normal_h2::alpha0derivs_taudelta (
             class(meos_normal_h2) this,
             real, intent(in) delta,
             real, intent(in) tau,
             real, dimension(0:2,0:2), intent(out) alp0 )
```

Parameters

out	*alp0*	alp0(i,j) = [(d_delta)^i(d_tau)^j alpha0]∗delta^i∗tau^j

6.74.1.2 alpharesderivs_taudelta()

```
procedure, public multiparameter_normal_h2::meos_normal_h2::alpharesderivs_taudelta (
             class(meos_normal_h2) this,
             real, intent(in) delta,
             real, intent(in) tau,
             real, dimension(0:2,0:2), intent(out) alpr )
```

Parameters

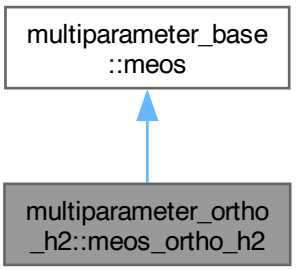
out	*alpr*	alpr(i,j) = (d_delta)^i(d_tau)^j alphaRes

The documentation for this type was generated from the following file:

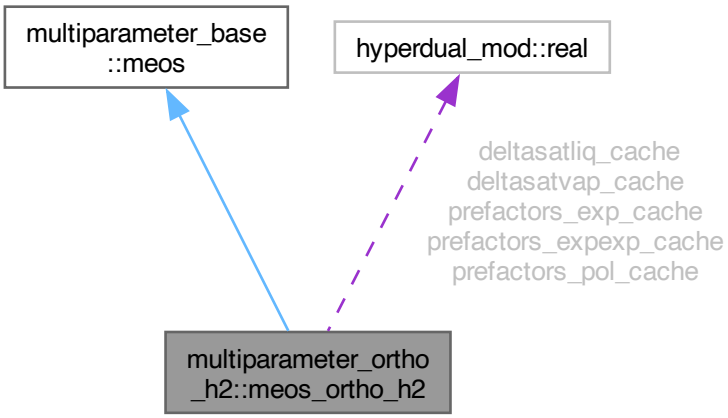
- multiparameter_normal_h2.f90

6.75 multiparameter_ortho_h2::meos_ortho_h2 Type Reference

Inheritance diagram for multiparameter_ortho_h2::meos_ortho_h2:



Collaboration diagram for multiparameter_ortho_h2::meos_ortho_h2:



Public Member Functions

- procedure, public alpha0derivs_taudelta (this, delta, tau, alp0)
- procedure, public alpharesderivs_taudelta (this, delta, tau, alpr)
- procedure, public **satdeltaestimate** (this, tau, phase)
- procedure, public **init** (this, use_rgas_fit)
- procedure, public **alpha0_hd_taudelta** (this, delta, tau)
- procedure, public **alphares_hd_taudelta** (this, delta, tau)
- procedure, pass, public **assign_meos** (this, other)

Public Member Functions inherited from multiparameter_base::meos

- procedure, public **mp_pressure** (this, rho, t, p, p_rho, p_t)

  Pressure and (optionally) its derivatives.

- procedure, public **alpha_to_f_conversion** (this, t, v, n, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- procedure, public **calc_f** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- procedure, public **calc_fid** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)

  Calculate reduced ideal Helmholtz energy and differentials.
- procedure, public **calc_zfac** (this, t, p, n, phase, z, z_t, z_p, z_n)
- procedure, public **calc_lnphi** (this, t, p, n, phase, lnphi, lnphi_t, lnphi_p, lnphi_n)
- procedure, public **calc_entropy** (this, t, p, n, phase, s, s_t, s_p, s_n, residual)
- procedure, public **calc_enthalpy** (this, t, p, n, phase, h, h_t, h_p, h_n, residual)
- procedure, public **calc_resgibbs** (this, t, p, n, phase, g, g_t, g_p, g_n)
- procedure, public densitysolver (this, t_spec, p_spec, phase_spec, rho, phase_found, ierr)
- procedure, public alphaderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, residual)
- procedure, public alpharesderivs_tv (this, t, v, alpr, alpr_t, alpr_v, alpr_tt, alpr_tv, alpr_vv, alpr_n, alpr_tn, alpr_vn, alpr_nn)
- procedure, public alphaidderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, alp_n, alp_tn, alp_vn, alp_nn)
- procedure, public **getcritpoint** (this, tcrit, pcrit, rhocrit)
- procedure(init_intf), deferred, public **init** (this, use_rgas_fit)

  Initiate compName, critical point and triple point.
- procedure, public cv (this, t, v)

  Isochoric heat capacity.
- procedure, public cp (this, t, v)

  Isobaric heat capacity.
- procedure, public speed_of_sound (this, t, v)

  [m/s]
- procedure, public **get_ref_state_spec** (this, ref_state, t, p, phase, solve)

  Get specification for the current reference state.
- procedure, public **set_ref_state** (this, t, p, v, h, s)

  Set reference state.
- procedure(satdeltaestimate_intf), deferred, public satdeltaestimate (this, tau, phase)

  An estimate delta_sat(tau_sat) for use in density solver.
- procedure(alpha0derivs_intf), deferred, public alpha0derivs_taudelta (this, delta, tau, alp0)

  [d^{j}alpha0/(d_tau)^j]∗tau^j
- procedure(alpharesderivs_intf), deferred, public alpharesderivs_taudelta (this, delta, tau, alpr)

  [d^{i+j}alphaRes/(d_delta)^i(d_tau)^j]∗delta^i∗tau^j
- procedure(alpha0_hd_intf), deferred, public alpha0_hd_taudelta (this, delta, tau)

  Calculate alpha0 using hyperdual numbers.
- procedure(alphares_hd_intf), deferred, public alphares_hd_taudelta (this, delta, tau)

  Calculate alphaRes using hyperdual numbers.
- procedure, public **assign_meos_base** (this, other)
- procedure(assign_meos_intf), deferred, pass, public **assign_meos** (this, other)
- generic, public **assignment** assign_meos

Public Attributes

- real **deltasatliq_cache**
- real **deltasatvap_cache**
- real, dimension(1:uppol) **prefactors_pol_cache**
- real, dimension(uppol+1:upexp) **prefactors_exp_cache**
- real, dimension(upexp+1:upexpexp) **prefactors_expexp_cache**

Public Attributes inherited from multiparameter_base::meos

- character(len=20), public **compname**

  Parameters in SI units. These are set in the deferred init routine.
- real, public **tc**
- real, public **pc**
- real, public **rc**
- real, public **acf**
- real, public **t_triple**
- real, public **p_triple**
- real, public **rholiq_triple**
- real, public **rhovap_triple**
- real, public **molarmass**

  (kg/mol)
- real, public **maxt**
- real, public **maxp**

  (K), (Pa)
- real, public **rgas_meos** = Rgas_default
- real, public **rgas_fit**

6.75.1 Member Function/Subroutine Documentation

6.75.1.1 alpha0derivs_taudelta()

```
procedure, public multiparameter_ortho_h2::meos_ortho_h2::alpha0derivs_taudelta (
             class(meos_ortho_h2) this,
             real, intent(in) delta,
             real, intent(in) tau,
             real, dimension(0:2,0:2), intent(out) alp0 )
```

Parameters

out	*alp0*	alp0(i,j) = [(d_delta)^i(d_tau)^j alpha0]∗delta^i∗tau^j

6.75.1.2 alpharesderivs_taudelta()

```
procedure, public multiparameter_ortho_h2::meos_ortho_h2::alpharesderivs_taudelta (
             class(meos_ortho_h2) this,
             real, intent(in) delta,
             real, intent(in) tau,
             real, dimension(0:2,0:2), intent(out) alpr )
```

Parameters

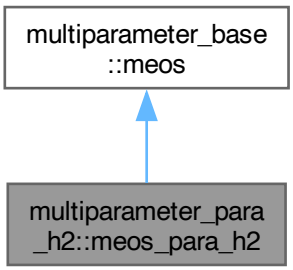
out	*alpr*	alpr(i,j) = (d_delta)^i(d_tau)^j alphaRes

The documentation for this type was generated from the following file:

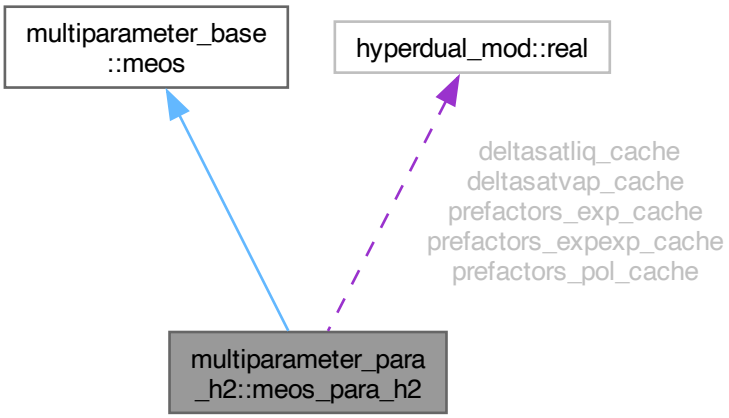
- multiparameter_ortho_h2.f90

6.76 multiparameter_para_h2::meos_para_h2 Type Reference

Inheritance diagram for multiparameter_para_h2::meos_para_h2:



Collaboration diagram for multiparameter_para_h2::meos_para_h2:



Public Member Functions

- procedure, public alpha0derivs_taudelta (this, delta, tau, alp0)
- procedure, public alpharesderivs_taudelta (this, delta, tau, alpr)
- procedure, public **satdeltaestimate** (this, tau, phase)
- procedure, public **init** (this, use_rgas_fit)
- procedure, public **alpha0_hd_taudelta** (this, delta, tau)
- procedure, public **alphares_hd_taudelta** (this, delta, tau)
- procedure, pass, public **assign_meos** (this, other)

Public Member Functions inherited from multiparameter_base::meos

- procedure, public **mp_pressure** (this, rho, t, p, p_rho, p_t)

  Pressure and (optionally) its derivatives.

- procedure, public **alpha_to_f_conversion** (this, t, v, n, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- procedure, public **calc_f** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- procedure, public **calc_fid** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)

  Calculate reduced ideal Helmholtz energy and differentials.
- procedure, public **calc_zfac** (this, t, p, n, phase, z, z_t, z_p, z_n)
- procedure, public **calc_lnphi** (this, t, p, n, phase, lnphi, lnphi_t, lnphi_p, lnphi_n)
- procedure, public **calc_entropy** (this, t, p, n, phase, s, s_t, s_p, s_n, residual)
- procedure, public **calc_enthalpy** (this, t, p, n, phase, h, h_t, h_p, h_n, residual)
- procedure, public **calc_resgibbs** (this, t, p, n, phase, g, g_t, g_p, g_n)
- procedure, public densitysolver (this, t_spec, p_spec, phase_spec, rho, phase_found, ierr)
- procedure, public alphaderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, residual)
- procedure, public alpharesderivs_tv (this, t, v, alpr, alpr_t, alpr_v, alpr_tt, alpr_tv, alpr_vv, alpr_n, alpr_tn, alpr_vn, alpr_nn)
- procedure, public alphaidderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, alp_n, alp_tn, alp_vn, alp_nn)
- procedure, public **getcritpoint** (this, tcrit, pcrit, rhocrit)
- procedure(init_intf), deferred, public **init** (this, use_rgas_fit)

  Initiate compName, critical point and triple point.
- procedure, public cv (this, t, v)

  Isochoric heat capacity.
- procedure, public cp (this, t, v)

  Isobaric heat capacity.
- procedure, public speed_of_sound (this, t, v)

  [m/s]
- procedure, public **get_ref_state_spec** (this, ref_state, t, p, phase, solve)

  Get specification for the current reference state.
- procedure, public **set_ref_state** (this, t, p, v, h, s)

  Set reference state.
- procedure(satdeltaestimate_intf), deferred, public satdeltaestimate (this, tau, phase)

  An estimate delta_sat(tau_sat) for use in density solver.
- procedure(alpha0derivs_intf), deferred, public alpha0derivs_taudelta (this, delta, tau, alp0)

  *[d^{j}alpha0/(d_tau)^j]*tau^j*
- procedure(alpharesderivs_intf), deferred, public alpharesderivs_taudelta (this, delta, tau, alpr)

  *[d^{i+j}alphaRes/(d_delta)^i(d_tau)^j]*delta^i*tau^j*
- procedure(alpha0_hd_intf), deferred, public alpha0_hd_taudelta (this, delta, tau)

  Calculate alpha0 using hyperdual numbers.
- procedure(alphares_hd_intf), deferred, public alphares_hd_taudelta (this, delta, tau)

  Calculate alphaRes using hyperdual numbers.
- procedure, public **assign_meos_base** (this, other)
- procedure(assign_meos_intf), deferred, pass, public **assign_meos** (this, other)
- generic, public **assignment** assign_meos

Public Attributes

- real **deltasatliq_cache**
- real **deltasatvap_cache**
- real, dimension(1:uppol) **prefactors_pol_cache**
- real, dimension(uppol+1:upexp) **prefactors_exp_cache**
- real, dimension(upexp+1:upexpexp) **prefactors_expexp_cache**

Public Attributes inherited from multiparameter_base::meos

- character(len=20), public **compname**

  Parameters in SI units. These are set in the deferred init routine.
- real, public **tc**
- real, public **pc**
- real, public **rc**
- real, public **acf**
- real, public **t_triple**
- real, public **p_triple**
- real, public **rholiq_triple**
- real, public **rhovap_triple**
- real, public **molarmass**

  (kg/mol)
- real, public **maxt**
- real, public **maxp**

  (K), (Pa)
- real, public **rgas_meos** = Rgas_default
- real, public **rgas_fit**

6.76.1 Member Function/Subroutine Documentation

6.76.1.1 alpha0derivs_taudelta()

```
procedure, public multiparameter_para_h2::meos_para_h2::alpha0derivs_taudelta (
             class(meos_para_h2) this,
             real, intent(in) delta,
             real, intent(in) tau,
             real, dimension(0:2,0:2), intent(out) alp0 )
```

Parameters

out	*alp0*	alp0(i,j) = [(d_delta)^i(d_tau)^j alpha0]∗delta^i∗tau^j

6.76.1.2 alpharesderivs_taudelta()

```
procedure, public multiparameter_para_h2::meos_para_h2::alpharesderivs_taudelta (
             class(meos_para_h2) this,
             real, intent(in) delta,
             real, intent(in) tau,
             real, dimension(0:2,0:2), intent(out) alpr )
```

Parameters

out	*alpr*	alpr(i,j) = (d_delta)^i(d_tau)^j alphaRes

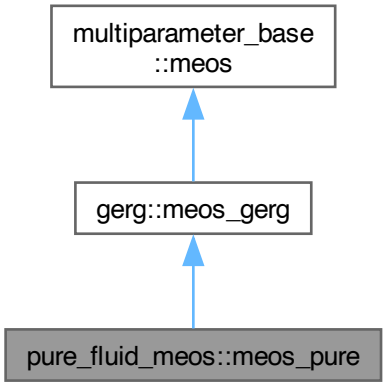
The documentation for this type was generated from the following file:

- multiparameter_para_h2.f90

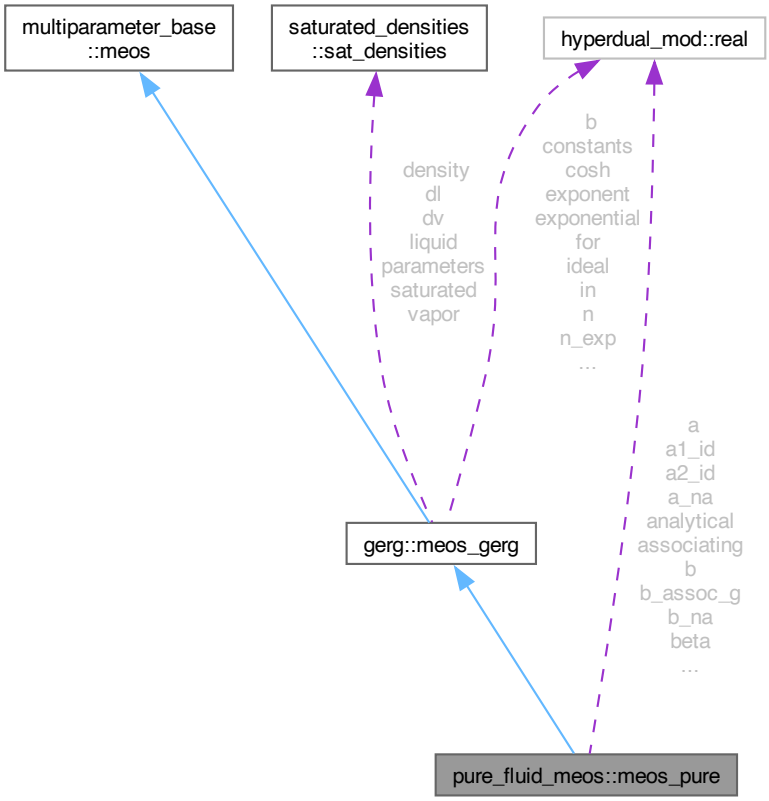
6.77 pure_fluid_meos::meos_pure Type Reference

Constructor for generic multiparameter equations of state.

Inheritance diagram for pure_fluid_meos::meos_pure:



Collaboration diagram for pure_fluid_meos::meos_pure:



Public Member Functions

- procedure, public alpha0derivs_taudelta (this, delta, tau, alp0)

  Specific reduced Helmholtz energy - ideal gas contributuion.
- procedure, public alpharesderivs_taudelta (this, delta, tau, alpr)

  Specific reduced residual Helmholtz energy.
- procedure, public **init** (this, use_rgas_fit)
- procedure, public **allocate_param** (this)
- procedure, public **alpha0_hd_taudelta** (this, delta, tau)

  Specific reduced Helmholtz energy - ideal gas contributuion. HYperdual numbers.
- procedure, public **alphares_hd_taudelta** (this, delta, tau)

  Specific reduced residual Helmholtz energy - hyperdual numbers.
- procedure, public **mp_pressure** (this, rho, t, p, p_rho, p_t)

  Pressure and (optionally) its derivatives.
- procedure, public **satdeltaestimate** (this, tau, phase)

  Provide initial guess for liquid and vapor densities.
- procedure, public **get_ref_state_spec** (this, ref_state, t, p, phase, solve)

  Get information on reference state.
- procedure, public **set_ref_state** (this, t, p, v, h, s)

  Set calculated reference state.
- procedure, pass, public **assign_meos** (this, other)

Public Member Functions inherited from gerg::meos_gerg

- procedure, public alpha0derivs_taudelta (this, delta, tau, alp0)

  Specific reduced Helmholtz energy - ideal gas contributuion.
- procedure, public alpharesderivs_taudelta (this, delta, tau, alpr)

  Specific residual reduced Helmholtz energy.
- procedure, public **satdeltaestimate** (this, tau, phase)

  Estimate saturated densities for density solver.
- procedure, public **init** (this, use_rgas_fit)
- procedure, public **allocate_param** (this)
- procedure, public **alpha0_hd_taudelta** (this, delta, tau)

  Specific reduced Helmholtz energy - ideal gas contributuion. Hyperdual numbers.
- procedure, public **alphares_hd_taudelta** (this, delta, tau)

  Specific residual reduced Helmholtz energy. Hyperdual numbers.
- procedure, pass, public **assign_meos** (this, other)

Public Member Functions inherited from multiparameter_base::meos

- procedure, public **mp_pressure** (this, rho, t, p, p_rho, p_t)

  Pressure and (optionally) its derivatives.
- procedure, public **alpha_to_f_conversion** (this, t, v, n, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- procedure, public **calc_f** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- procedure, public **calc_fid** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)

  Calculate reduced ideal Helmholtz energy and differentials.
- procedure, public **calc_zfac** (this, t, p, n, phase, z, z_t, z_p, z_n)
- procedure, public **calc_lnphi** (this, t, p, n, phase, lnphi, lnphi_t, lnphi_p, lnphi_n)
- procedure, public **calc_entropy** (this, t, p, n, phase, s, s_t, s_p, s_n, residual)
- procedure, public **calc_enthalpy** (this, t, p, n, phase, h, h_t, h_p, h_n, residual)
- procedure, public **calc_resgibbs** (this, t, p, n, phase, g, g_t, g_p, g_n)
- procedure, public densitysolver (this, t_spec, p_spec, phase_spec, rho, phase_found, ierr)

- procedure, public alphaderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, residual)
- procedure, public alpharesderivs_tv (this, t, v, alpr, alpr_t, alpr_v, alpr_tt, alpr_tv, alpr_vv, alpr_n, alpr_tn, alpr_vn, alpr_nn)
- procedure, public alphaidderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, alp_n, alp_tn, alp_vn, alp_nn)
- procedure, public **getcritpoint** (this, tcrit, pcrit, rhocrit)
- procedure(init_intf), deferred, public **init** (this, use_rgas_fit)

  Initiate compName, critical point and triple point.
- procedure, public cv (this, t, v)

  Isochoric heat capacity.
- procedure, public cp (this, t, v)

  Isobaric heat capacity.
- procedure, public speed_of_sound (this, t, v)

  [m/s]
- procedure, public **get_ref_state_spec** (this, ref_state, t, p, phase, solve)

  Get specification for the current reference state.
- procedure, public **set_ref_state** (this, t, p, v, h, s)

  Set reference state.
- procedure(satdeltaestimate_intf), deferred, public satdeltaestimate (this, tau, phase)

  An estimate delta_sat(tau_sat) for use in density solver.
- procedure(alpha0derivs_intf), deferred, public alpha0derivs_taudelta (this, delta, tau, alp0)

  *[d^{j}alpha0/(d_tau)^j]*tau^j*
- procedure(alpharesderivs_intf), deferred, public alpharesderivs_taudelta (this, delta, tau, alpr)

  *[d^{i+j}alphaRes/(d_delta)^i(d_tau)^j]*delta^i*tau^j*
- procedure(alpha0_hd_intf), deferred, public alpha0_hd_taudelta (this, delta, tau)

  Calculate alpha0 using hyperdual numbers.
- procedure(alphares_hd_intf), deferred, public alphares_hd_taudelta (this, delta, tau)

  Calculate alphaRes using hyperdual numbers.
- procedure, public **assign_meos_base** (this, other)
- procedure(assign_meos_intf), deferred, pass, public **assign_meos** (this, other)
- generic, public **assignment** assign_meos

Public Attributes

- real **t_nbp** = 0.0
- real **normal**
- real **boiling**
- real **point**
- real **temperature**
- character(len=comp_name_len) **ref_state**
- character(len=comp_name_len) **reference**
- character(len=comp_name_len) **state**
- integer **n1_id** = 0
- integer **number**
- integer, dimension(:), allocatable **of**
- integer, dimension(:), allocatable **polynomial**
- integer, dimension(:), allocatable **terms**
- integer, dimension(:), allocatable **in**
- integer **ideal**
- integer **gas**
- integer **cp**
- integer **n_id** = 0
- integer **and**

- integer **einstein**
- real **a1_id** = 0
- real **integration**
- real **constant**
- real **entropy**
- real **a2_id** = 0
- real **enthalpy**
- real, dimension(:), allocatable **c_id**
- real, dimension(:), allocatable **prefactor**
- real, dimension(:), allocatable **parameter**
- real, dimension(:), allocatable **t_id**
- real, dimension(:), allocatable **tau**
- real, dimension(:), allocatable **exponent**
- integer **n_gauss** = 0
- integer, dimension(:), allocatable **gauss**
- integer **n_nona** = 0
- integer, dimension(:), allocatable **non**
- integer, dimension(:), allocatable **analytical**
- integer **n_assoc** = 0
- integer **association**
- real, dimension(:), allocatable **g_exp**
- real, dimension(:), allocatable **scaling**
- real, dimension(:), allocatable **inside**
- real, dimension(:), allocatable **exponential**
- real, dimension(:), allocatable **term**
- real, dimension(:), allocatable **n_g**
- real, dimension(:), allocatable **for**
- real, dimension(:), allocatable **gauss**
- real, dimension(:), allocatable **terms**
- real, dimension(:), allocatable **t_g**
- real, dimension(:), allocatable **polynomial**
- real, dimension(:), allocatable **in**
- integer, dimension(:), allocatable **d_g**
- integer, dimension(:), allocatable **delta**
- integer, dimension(:), allocatable **exponent**
- integer, dimension(:), allocatable **for**
- integer, dimension(:), allocatable **prefactor**
- real, dimension(:), allocatable **eta_g**
- real, dimension(:), allocatable **delta**
- real, dimension(:), allocatable **function**
- real, dimension(:), allocatable **of**
- real, dimension(:), allocatable **the**
- real, dimension(:), allocatable **beta_g**
- real, dimension(:), allocatable **gamma_g**
- real, dimension(:), allocatable **offset**
- real, dimension(:), allocatable **epsilon_g**
- integer, dimension(:), allocatable **tauexp_g**
- integer, dimension(:), allocatable **tau**
- integer, dimension(:), allocatable **term**
- integer, dimension(:), allocatable **inside**
- integer, dimension(:), allocatable **exponential**
- integer, dimension(:), allocatable **function**
- integer, dimension(:), allocatable **the**
- integer, dimension(:), allocatable **delexp_g**
- real, dimension(:), allocatable **b_assoc_g**

- real, dimension(:), allocatable **b**
- real, dimension(:), allocatable **modefied**
- real, dimension(:), allocatable **used**
- real, dimension(:), allocatable **with**
- real, dimension(:), allocatable **associating**
- real, dimension(:), allocatable **fluids**
- real, dimension(:), allocatable **n_na**
- real, dimension(:), allocatable **non**
- real, dimension(:), allocatable **analytical**
- real, dimension(:), allocatable **a_na**
- real, dimension(:), allocatable **a**
- real, dimension(:), allocatable **b_na**
- real, dimension(:), allocatable **beta_na**
- real, dimension(:), allocatable **beta**
- real, dimension(:), allocatable **big_a_na**
- real, dimension(:), allocatable **big_b_na**
- real, dimension(:), allocatable **big_c_na**
- real, dimension(:), allocatable **c**
- real, dimension(:), allocatable **big_d_na**
- real, dimension(:), allocatable **d**

Public Attributes inherited from gerg::meos_gerg

- integer **i_comp** = 0
- integer **index**
- integer, dimension(:), allocatable **of**
- integer **component**
- integer, dimension(:), allocatable **in**
- integer **gergdatadb**
- integer **n_cosh** = 0
- integer **number**
- integer, dimension(:), allocatable **cosh**
- integer, dimension(:), allocatable **terms**
- integer **n_sinh** = 0
- integer **and**
- integer, dimension(:), allocatable **sinh**
- real, dimension(3) **n**
- real **constants**
- real, dimension(:), allocatable **v**
- real, dimension(:), allocatable **prefactor**
- real, dimension(:), allocatable **ideal**
- real, dimension(:), allocatable **terms**
- real, dimension(:), allocatable **b**
- real, dimension(:), allocatable **cosh**
- real, dimension(:), allocatable **sinh**
- real, dimension(:), allocatable **parameter**
- real, dimension(:), allocatable **for**
- integer **uppol** = 0
- integer, dimension(:), allocatable **polynomial**
- integer **upexp** = 0
- integer, dimension(:), allocatable **exponential**
- real, dimension(:), allocatable **n_pol**
- real, dimension(:), allocatable **polynomial**
- real, dimension(:), allocatable **n_exp**
- real, dimension(:), allocatable **exponential**

- real, dimension(:), allocatable **t_pol**
- real, dimension(:), allocatable **tau**
- real, dimension(:), allocatable **exponent**
- real, dimension(:), allocatable **t_exp**
- real, dimension(:), allocatable **in**
- integer, dimension(:), allocatable **d_pol**
- integer, dimension(:), allocatable **delta**
- integer, dimension(:), allocatable **exponent**
- integer, dimension(:), allocatable **d_exp**
- integer, dimension(:), allocatable **for**
- integer, dimension(:), allocatable **prefactor**
- integer, dimension(:), allocatable **l_exp**
- integer, dimension(:), allocatable **term**
- type(sat_densities) **dl**
- type(sat_densities) **parameters**
- type(sat_densities) **liquid**
- type(sat_densities) **saturated**
- type(sat_densities) **density**
- type(sat_densities) **dv**
- type(sat_densities) **vapor**

Public Attributes inherited from multiparameter_base::meos

- character(len=20), public **compname**

  Parameters in SI units. These are set in the deferred init routine.
- real, public **tc**
- real, public **pc**
- real, public **rc**
- real, public **acf**
- real, public **t_triple**
- real, public **p_triple**
- real, public **rholiq_triple**
- real, public **rhovap_triple**
- real, public **molarmass**

  (kg/mol)
- real, public **maxt**
- real, public **maxp**

  (K), (Pa)
- real, public **rgas_meos** = Rgas_default
- real, public **rgas_fit**

6.77.1 Detailed Description

Constructor for generic multiparameter equations of state.

6.77.2 Member Function/Subroutine Documentation

6.77.2.1 alpha0derivs_taudelta()

```
procedure, public pure_fluid_meos::meos_pure::alpha0derivs_taudelta (
             class(meos_pure) this,
             real, intent(in) delta,
             real, intent(in) tau,
             real, dimension(0:2,0:2), intent(out) alp0 )
```

Specific reduced Helmholtz energy - ideal gas contributuion.

Parameters

out	*alp0*	alp0(i,j) = [(d_delta)^i(d_tau)^j alpha0]*delta^i*tau^j

6.77.2.2 alpharesderivs_taudelta()

```
procedure, public pure_fluid_meos::meos_pure::alpharesderivs_taudelta (
            class(meos_pure) this,
            real, intent(in) delta,
            real, intent(in) tau,
            real, dimension(0:2,0:2), intent(out) alpr )
```

Specific reduced residual Helmholtz energy.

Parameters

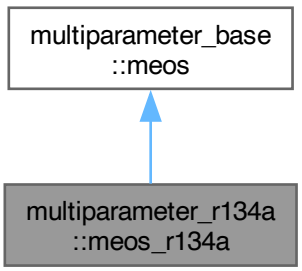
out	*alpr*	alpr(i,j) = (d_delta)^i(d_tau)^j alphaRes

The documentation for this type was generated from the following file:

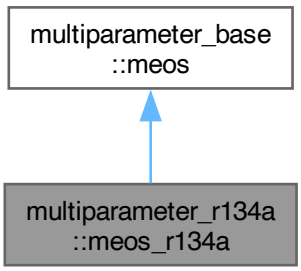
- pure_fluid_meos.f90

6.78 multiparameter_r134a::meos_r134a Type Reference

R134A multiparameter equation of state (R. Tillner-Roth and H. Dieter Baehr).
Inheritance diagram for multiparameter_r134a::meos_r134a:



Collaboration diagram for multiparameter_r134a::meos_r134a:



Public Member Functions

- procedure, public alpha0derivs_taudelta (this, delta, tau, alp0)
- procedure, public alpharesderivs_taudelta (this, delta, tau, alpr)
- procedure, public **satdeltaestimate** (this, tau, phase)
- procedure, public **init** (this, use_rgas_fit)
- procedure, public **alpha0_hd_taudelta** (this, delta, tau)
- procedure, public **alphares_hd_taudelta** (this, delta, tau)
- procedure, pass, public **assign_meos** (this, other)

Public Member Functions inherited from multiparameter_base::meos

- procedure, public **mp_pressure** (this, rho, t, p, p_rho, p_t)

  Pressure and (optionally) its derivatives.
- procedure, public **alpha_to_f_conversion** (this, t, v, n, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- procedure, public **calc_f** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)
- procedure, public **calc_fid** (this, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)

  Calculate reduced ideal Helmholtz energy and differentials.
- procedure, public **calc_zfac** (this, t, p, n, phase, z, z_t, z_p, z_n)
- procedure, public **calc_lnphi** (this, t, p, n, phase, lnphi, lnphi_t, lnphi_p, lnphi_n)
- procedure, public **calc_entropy** (this, t, p, n, phase, s, s_t, s_p, s_n, residual)
- procedure, public **calc_enthalpy** (this, t, p, n, phase, h, h_t, h_p, h_n, residual)
- procedure, public **calc_resgibbs** (this, t, p, n, phase, g, g_t, g_p, g_n)
- procedure, public densitysolver (this, t_spec, p_spec, phase_spec, rho, phase_found, ierr)
- procedure, public alphaderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, residual)
- procedure, public alpharesderivs_tv (this, t, v, alpr, alpr_t, alpr_v, alpr_tt, alpr_tv, alpr_vv, alpr_n, alpr_tn, alpr_vn, alpr_nn)
- procedure, public alphaidderivs_tv (this, t, v, alp, alp_t, alp_v, alp_tt, alp_tv, alp_vv, alp_n, alp_tn, alp_vn, alp_nn)
- procedure, public **getcritpoint** (this, tcrit, pcrit, rhocrit)
- procedure(init_intf), deferred, public **init** (this, use_rgas_fit)

  Initiate compName, critical point and triple point.
- procedure, public cv (this, t, v)

  Isochoric heat capacity.
- procedure, public cp (this, t, v)

  Isobaric heat capacity.

- procedure, public speed_of_sound (this, t, v)

  [m/s]
- procedure, public **get_ref_state_spec** (this, ref_state, t, p, phase, solve)

  Get specification for the current reference state.
- procedure, public **set_ref_state** (this, t, p, v, h, s)

  Set reference state.
- procedure(satdeltaestimate_intf), deferred, public satdeltaestimate (this, tau, phase)

  An estimate delta_sat(tau_sat) for use in density solver.
- procedure(alpha0derivs_intf), deferred, public alpha0derivs_taudelta (this, delta, tau, alp0)

  *[d^{j}alpha0/(d_tau)^j]*tau^j*
- procedure(alpharesderivs_intf), deferred, public alpharesderivs_taudelta (this, delta, tau, alpr)

  *[d^{i+j}alphaRes/(d_delta)^i(d_tau)^j]*delta^i*tau^j*
- procedure(alpha0_hd_intf), deferred, public alpha0_hd_taudelta (this, delta, tau)

  Calculate alpha0 using hyperdual numbers.
- procedure(alphares_hd_intf), deferred, public alphares_hd_taudelta (this, delta, tau)

  Calculate alphaRes using hyperdual numbers.
- procedure, public **assign_meos_base** (this, other)
- procedure(assign_meos_intf), deferred, pass, public **assign_meos** (this, other)
- generic, public **assignment** assign_meos

Additional Inherited Members

Public Attributes inherited from multiparameter_base::meos

- character(len=20), public **compname**

  Parameters in SI units. These are set in the deferred init routine.
- real, public **tc**
- real, public **pc**
- real, public **rc**
- real, public **acf**
- real, public **t_triple**
- real, public **p_triple**
- real, public **rholiq_triple**
- real, public **rhovap_triple**
- real, public **molarmass**

  (kg/mol)
- real, public **maxt**
- real, public **maxp**

  (K), (Pa)
- real, public **rgas_meos** = Rgas_default
- real, public **rgas_fit**

6.78.1 Detailed Description

R134A multiparameter equation of state (R. Tillner-Roth and H. Dieter Baehr).

6.78.2 Member Function/Subroutine Documentation

6.78.2.1 alpha0derivs_taudelta()

```
procedure, public multiparameter_r134a::meos_r134a::alpha0derivs_taudelta (
             class(meos_r134a) this,
             real, intent(in) delta,
             real, intent(in) tau,
             real, dimension(0:2,0:2), intent(out) alp0 )
```

Parameters

out	*alp0*	alp0(i,j) = [(d_delta)^i(d_tau)^j alpha0]∗delta^i∗tau^j

6.78.2.2 alpharesderivs_taudelta()

```
procedure, public multiparameter_r134a::meos_r134a::alpharesderivs_taudelta (
             class(meos_r134a) this,
             real, intent(in) delta,
             real, intent(in) tau,
             real, dimension(0:2,0:2), intent(out) alpr )
```

Parameters

out	*alpr*	alpr(i,j) = (d_delta)^i(d_tau)^j alphaRes

The documentation for this type was generated from the following file:

- multiparameter_r134a.f90

6.79 meosdatadb::meosdata Type Reference

Public Attributes

- character(len=uid_len) **ident**

  The component ID.
- character(len=comp_name_len) **name**

  The component name.
- character(len=comp_name_len) **default_ref_state**

  The default reference state.
- character(len=bibref_len) **bibref**

  Digital Object Identifier (DOI) or reference.
- real **mw**

  Mole weight (g/mol)
- real **ttr**

  Triple point temperature (K)
- real **ptr**

  Triple point pressure (kPa)
- real **t_nbp**

  Normal boiling point temperature (K)
- real **tc**

  Critical temperature (K)
- real **pc**

  Critical pressure (kPa)
- real **rhoc**

  Critical density (mol/l)
- real **tr**

  Reducing temperature (K)
- real **rhor**

  Reducing density (mol/l)
- real **rgas**

  Gas constant (J/mol-K)

- real **acf**

  Acentric factor.
- real **t_max**

  Maximum temperature (K)
- real **p_max**

  Maximum pressure (kPa)
- integer **n_poly_eos**

  Number of polynomial terms.
- integer **n_exp_eos**

  Number of exponential terms.
- integer **n_gauss_eos**

  Number of Gaussian bell-shaped terms.
- integer **n_nona_eos**

  Number of non-analytical terms.
- integer **n_assoc_eos**

  Number of association terms.
- real, dimension(meos_max_n) **n_eos**

  Prefactor.
- real, dimension(meos_max_n) **t_eos**

  Tau exponent.
- integer, dimension(meos_max_n) **d_eos**

  Delta exponent.
- integer, dimension(meos_max_n) **l_eos**

  Exponential delta exponent.
- real, dimension(meos_max_n) **g_eos**

  Exponential delta prefactor.
- real, dimension(meos_max_n_gauss) **eta_eos**

  Prefactor delta term.
- real, dimension(meos_max_n_gauss) **beta_eos**

  Prefactor tau term.
- real, dimension(meos_max_n_gauss) **gamma_eos**

  Tau correction.
- real, dimension(meos_max_n_gauss) **epsilon_eos**

  Delta correction.
- integer, dimension(meos_max_n_gauss) **tau_exp_eos**

  Tau term exponent.
- integer, dimension(meos_max_n_gauss) **del_exp_eos**

  Delta term exponent.
- real, dimension(2) **b_assoc_eos**

  Association b.
- real, dimension(meos_max_n_nona) **n_na**

  Prefactor non-analytical terms.
- real, dimension(meos_max_n_nona) **a_na**

  Non-analytical parameter a.
- real, dimension(meos_max_n_nona) **b_na**

  Non-analytical parameter b.
- real, dimension(meos_max_n_nona) **beta_na**

  Non-analytical parameter beta.
- real, dimension(meos_max_n_nona) **big_a_na**

  Non-analytical parameter A.
- real, dimension(meos_max_n_nona) **big_b_na**

Non-analytical parameter B.

- real, dimension(meos_max_n_nona) **big_c_na**

  Non-analytical parameter C.

- real, dimension(meos_max_n_nona) **big_d_na**

  Non-analytical parameter D.

- integer **n1_id**

  Number of polynomial terms.

- integer **n_id**

  Overall number of ID terms.

- real, dimension(meos_id_max_n) **c_id**

  Polynomial parameter/Einstein function parameter.

- real, dimension(meos_id_max_n) **t_id**

  Einstein function exp-parameter.

- real **a1_id**

  Integration constant for entropy.

- real **a2_id**

  Integration constant for enthalpy.

The documentation for this type was generated from the following file:

- meosdatadb.f90

6.80 saftvrmie_datadb::miekijdata Type Reference

INTERACTION PARAMETERS FOR THE SAFT-VR-MIE DISPERSION TERM.

Public Attributes

- integer **eosidx**
- character(len=uid_len) **uid1**
- character(len=uid_len) **uid2**
- character(len=ref_len) **ref**
- character(len=bibref_len) **bib_ref**
- real **kijvalue**

6.80.1 Detailed Description

INTERACTION PARAMETERS FOR THE SAFT-VR-MIE DISPERSION TERM.
The documentation for this type was generated from the following file:

- saftvrmie_datadb.f90

6.81 hyperdual_mod::min Interface Reference

Public Member Functions

- elemental type(hyperdual) function **min_dd** (v1, v2)
- elemental type(hyperdual) function **min_dr** (v1, v2)
- elemental type(hyperdual) function **min_rd** (v1, v2)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.82 cubic_eos::mix_label_mapping Type Reference

Public Attributes

- integer **mix_idx_group**
- integer **mix_idx**
- character(len=short_label_len) **short_label**
- character(len=label_len) **label**
- character(len=label_len) **alias**

The documentation for this type was generated from the following file:

- cubic_eos.f90

6.83 cubic_eos::mixexcessgibbs Type Reference

Public Member Functions

- procedure, public **dealloc** (mixge)
- procedure, public **excess_gibbs_allocate_and_init** (mixge, nc)
- procedure **assign_excess_gibbs_mix** (mixge1, mixge2)
- generic, public **assignment** (mixge1, mixge2)

Public Attributes

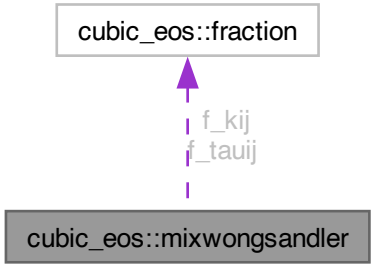
- integer **mge**
- integer, dimension(:,:), allocatable **correlation**
- real, dimension(:,:), allocatable **alpha**
- real, dimension(:,:), allocatable **age**
- real, dimension(:,:), allocatable **bge**
- real, dimension(:,:), allocatable **cge**

The documentation for this type was generated from the following file:

- cubic_eos.f90

6.84 cubic_eos::mixwongsandler Type Reference

Collaboration diagram for cubic_eos::mixwongsandler:



Public Member Functions

- procedure, public **dealloc** (mixws)
- procedure, public **ws_allocate_and_init** (mixws, nc)
- procedure **assign_ws_mix** (mixws1, mixws2)
- generic, public **assignment** (mixws1, mixws2)

Public Attributes

- real, dimension(:,:), allocatable **alphaij**
- type(fraction), dimension(:,:), allocatable **f_kij**
- type(fraction), dimension(:,:), allocatable **f_tauij**

The documentation for this type was generated from the following file:

- cubic_eos.f90

6.85 multipol_var::multipol_param Type Reference

Public Member Functions

- procedure, public init_multipol_param (self, nce, mu, q, m, sigma_ij, eps_divk_ij)

Public Attributes

- real, dimension(:), allocatable **mu_star_2**

  Reduced dipol moment squared [-].
- real, dimension(:), allocatable **q_star_2**

  Reduced quadrupol moment squared [-].
- integer, dimension(:), allocatable **l_mu**

  Number of dipol moments per molecule.
- integer, dimension(:), allocatable **l_q**

  Number of quadrupol moments per molecule.
- real, dimension(:), allocatable **m**
- real, dimension(:,:), allocatable **eps_divk_ij**
- real, dimension(:,:), allocatable **sigma_ij**
- real, dimension(:,:), allocatable **sigma_ij_3**
- real, dimension(:,:), allocatable **sigma_ij_5**
- real, dimension(:,:,:), allocatable **a_ij_qq**
- real, dimension(:,:,:), allocatable **b_ij_qq**
- real, dimension(:,:,:,:), allocatable **c_ijk_qq**
- real, dimension(:,:,:), allocatable **a_ij_dd**
- real, dimension(:,:,:), allocatable **b_ij_dd**
- real, dimension(:,:,:,:), allocatable **c_ijk_dd**
- real, dimension(:,:,:), allocatable **a_ij_dq**
- real, dimension(:,:,:), allocatable **b_ij_dq**
- real, dimension(:,:,:,:), allocatable **c_ijk_dq**
- integer, dimension(:), allocatable **mu_indices**
- integer, dimension(:), allocatable **q_indices**
- integer **num_mu**

  Number of diplo moments different from zero.
- integer **num_q**

  Number of diplo moments different from zero.
- logical **enable_qq** = .true.
- logical **enable_dd** = .true.
- logical **enable_dq** = .true.

6.85.1 Member Function/Subroutine Documentation

6.85.1.1 init_multipol_param()

```
procedure, public multipol_var::multipol_param::init_multipol_param (
             class(multipol_param), intent(inout) self,
             integer, intent(in) nce,
             real, dimension(nce), intent(in) mu,
             real, dimension(nce), intent(in) q,
             real, dimension(nce), intent(in) m,
             real, dimension(nce,nce), intent(in) sigma_ij,
             real, dimension(nce,nce), intent(in) eps_divk_ij )
```

Parameters

in	*mu*	[D]
in	*q*	[ÃåD]

The documentation for this type was generated from the following file:

- multipol_var.f90

6.86 mbwr::nijlarray Type Reference

Public Attributes

- integer **len**
- real, dimension(:), allocatable **n**
- integer, dimension(:), allocatable **i**
- real, dimension(:), allocatable **j**
- integer, dimension(:), allocatable **l**
- real **gamma**

The documentation for this type was generated from the following file:

- mbwr.f90

6.87 hyperdual_mod::nint Interface Reference

Public Member Functions

- elemental integer function **ninthyperdual** (v1)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.88 multiparameter_base::nist_meos_ptr Type Reference

Public Attributes

- class(meos), pointer **meos**

The documentation for this type was generated from the following file:

- multiparameter_base.f90

6.89 nonlinear_solvers::nonlinear_solver Type Reference

A type that contains solver information.

Public Attributes

- integer **isolver** = NS_NEWTON_LS

  Solver choice.
- integer **max_it** = 100

  Maximum number of iterations, if applicable.
- integer **iter** = 0

  Number of iterations to solution.
- real **rel_tol** = 1e-20

  Relative tolerance.
- real **abs_tol** = 1e-10

  Absolute tolerance.
- logical **analyt_jac** = .true.

  Whether the Jacobian should be computed analytically with a user-provided procedure, or approximated using approximate_jacobian.
- integer **analyt_jac_order** = 1
- logical **analyt_hess** = .false.

  Whether the inverse of the Hessian should be computed analytically with a user-provided procedure, or approximated by inverting the Jacobian.
- logical **verbose** = .false.

  Whether to output information about the solution.
- integer **ls_max_it** = 10

  Number of line search trials.
- logical **limit_x_values** = .true.

  Limit x to make sure: xmin <= x <= xmax.
- logical **symmetric_jac** = .false.

  Is the Jacobean symmetric?
- integer **exitflag** = 0

  Status flag. 0 - Solver converged 1 - No solution found after max_it iterations 2 - Could not invert jacobian 3 - Floating point error, probably divergence.
- real **error_on_exit** = 0.0

6.89.1 Detailed Description

A type that contains solver information.
The documentation for this type was generated from the following file:

- nonlinear_solvers.f90

6.90 hyperdual_mod::operator(∗) Interface Reference

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.91 hyperdual_mod::operator(∗∗) Interface Reference

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.92 hyperdual_mod::operator(+) Interface Reference

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.93 hyperdual_mod::operator(-) Interface Reference

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.94 hyperdual_mod::operator(.eq.) Interface Reference

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.95 hyperdual_mod::operator(.ge.) Interface Reference

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.96 hyperdual_mod::operator(.gt.) Interface Reference

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.97 hyperdual_mod::operator(.le.) Interface Reference

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.98 hyperdual_mod::operator(.lt.) Interface Reference

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.99 hyperdual_mod::operator(.ne.) Interface Reference

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.100 hyperdual_mod::operator(/) Interface Reference

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.101 optimizers::optim_param Type Reference

A type that contains solver information.

Public Attributes

- integer **ioptim** = NO_MOD_NEWTON

  Optimizer choice.
- integer **max_iter** = 1000

  Maximum number of iterations, if applicable.
- real **rel_tol** = 1.0e5∗machine_prec

  Relative tolerance.
- real **wolfe** = 1.0e-3

  Line search parameter.
- integer **max_line_search_iter** = 10

  Number of line search trials.
- logical **gradient_termination** = .false.

  Terminate besed on gradient infinity norm.
- logical **line_search_control** = .false.

  Turn off/on line search control (line search controlled by last param element)
- integer **iter** = 0

  Number of iterations to solution.
- real **of** = 0.0

  Objective function value at return.
- real **error** = 0.0

  Error at return.
- integer **exitflag** = 0

  Status flag. 0 - Solver converged 1 - No solution found after max_it iterations -1 - Premature return.
- logical **testeigenvalues** = .false.

  Test eigenvalues of the Jacobean.

6.101.1 Detailed Description

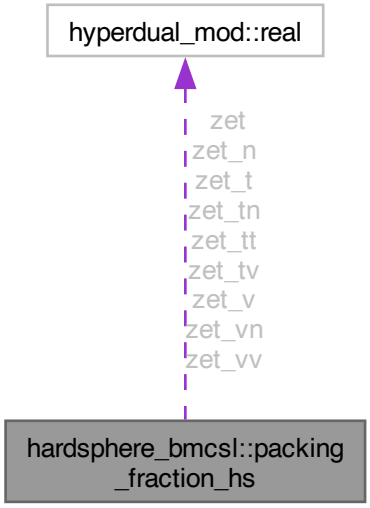
A type that contains solver information.
The documentation for this type was generated from the following file:

- optimizer.f90

6.102 hardsphere_bmcsl::packing_fraction_hs Type Reference

Container for zeta's (0, 1, 2, 3 and mu). These are moments of the number density.

Collaboration diagram for hardsphere_bmcsl::packing_fraction_hs:



Public Member Functions

- procedure, public **allocate** (zeta, nc)

  Allocated zetaeta memory and initialize to zero.
- procedure, public **deallocate** (zeta)

  Free allocated packing_fraction_hs memory.
- procedure, public **assign_packing_fraction_hs** (this, other)
- generic, public **assignment** (this, other)

Public Attributes

- real, dimension(5) **zet**

  Moments of the number density.
- real, dimension(5) **zet_t**

  Temperature differential of the moments of the number density and mu.
- real, dimension(5) **zet_tt**

  Second temperature differential of the moments of the number density and mu.
- real, dimension(5) **zet_v**

  Volume differential of the moments of the number density and mu.
- real, dimension(5) **zet_vv**

  Second volume differential of the moments of the number density and mu.
- real, dimension(5) **zet_tv**

  Temperature and volume differential of the moments of the number density and mu.
- real, dimension(:,:), allocatable **zet_n**

  Mol number differential of the moments of the number density and mu.
- real, dimension(:,:), allocatable **zet_vn**

  Mol number and volume differential of the moments of the number density and mu.
- real, dimension(:,:), allocatable **zet_tn**

  Mol number and temperature differential of the moments of the number density and mu.

6.102.1 Detailed Description

Container for zeta's (0, 1, 2, 3 and mu). These are moments of the number density.
The documentation for this type was generated from the following file:

- hardsphere_bmcsl.f90

6.103 pc_saft_datadb::pc_saft_data Type Reference

PURE COMPONENT PARAMETERS. This data structure stores pure component parameters for the PC-SAFT equation of state.

Public Attributes

- integer **eosidx**
- character(len=uid_len) **compname**
- real **m**

  [-]. Mean number of segments.
- real **sigma**

  [m]. Temperature-independent segment diameter.
- real **eps_depth_divk**

  [K]. Well depth divided by Boltzmann's c.
- real **eps**

  [J/mol].
- real **beta**

  [-]. Also known as kappa in SAFT literature.
- integer **assoc_scheme**

  Association scheme.
- real **mu**

  Dipole-moment [D].
- real **q**

  Quadrupol-moment [ÃåD].
- character(len=bibref_len) **bib_ref**
- character(len=ref_len) **ref**

6.103.1 Detailed Description

PURE COMPONENT PARAMETERS. This data structure stores pure component parameters for the PC-SAFT equation of state.
The documentation for this type was generated from the following file:

- pc_saft_datadb.f90

6.104 pc_saft_datadb::pckijdata Type Reference

TEMPERATURE-INDEPENDENT INTERACTION PARAMETERS FOR PC-SAFT DISPERSION TERM.

Public Attributes

- integer **eosidx**
- character(len=uid_len) **uid1**
- character(len=uid_len) **uid2**
- character(len=ref_len) **ref**
- character(len=bibref_len) **bib_ref**
- real **kijvalue**
- integer **eps_comb_rule**
- integer **beta_comb_rule**

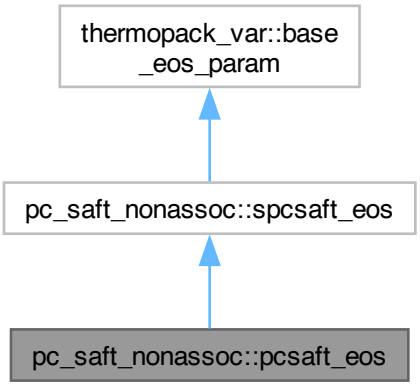
6.104.1 Detailed Description

TEMPERATURE-INDEPENDENT INTERACTION PARAMETERS FOR PC-SAFT DISPERSION TERM.
The documentation for this type was generated from the following file:

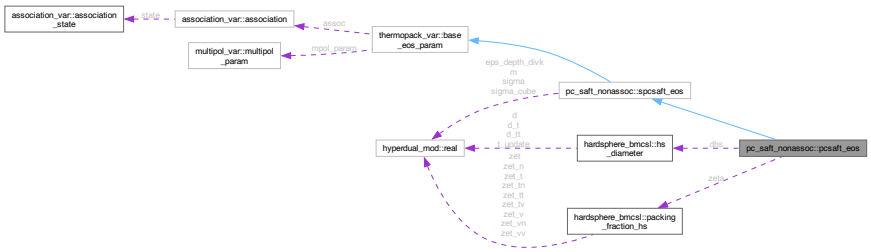
- pc_saft_datadb.f90

6.105 pc_saft_nonassoc::pcsaft_eos Type Reference

Inheritance diagram for pc_saft_nonassoc::pcsaft_eos:



Collaboration diagram for pc_saft_nonassoc::pcsaft_eos:



Public Member Functions

- procedure, public **dealloc** (eos)
- procedure, public allocate_and_init (eos, nc, eos_label)
- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from pc_saft_nonassoc::spcsaft_eos

- procedure, public **dealloc** (eos)
- procedure, public allocate_and_init (eos, nc, eos_label)
- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from thermopack_var::base_eos_param

- procedure(allocate_and_init_intf), deferred, public allocate_and_init (eos, nc, eos_label)
- procedure, public **dealloc** (eos)
- procedure(assign_intf), deferred, pass, public **assign_eos** (this, other)
- generic, public **assignment** assign_eos
- procedure, public **assign_base_eos_param** (this, other)

Public Attributes

- type(hs_diameter) **dhs**
- type(packing_fraction_hs) **zeta**

Public Attributes inherited from pc_saft_nonassoc::spcsaft_eos

- real, dimension(:), allocatable **m**

  [-]
- real, dimension(:,:), allocatable **sigma**

  [m]
- real, dimension(:,:), allocatable **eps_depth_divk**

  [K]
- real, dimension(:,:), allocatable **sigma_cube**

  [m^3]

Public Attributes inherited from thermopack_var::base_eos_param

- character(len=eosid_len) **eosid**

  Eos identifier.
- integer **eosidx**

  Eos group index.
- integer **subeosidx**

  Eos sub-index.
- integer **volumeshiftid** = 0

  0: No volume shift, 1:Peneloux shift
- logical **iselectrolyteeos** = .false.

  Used to enable electrolytes.
- type(association), pointer **assoc** => NULL()
- type(multipol_param), pointer **mpol_param** => NULL()

6.105.1 Member Function/Subroutine Documentation

6.105.1.1 allocate_and_init()

```
procedure, public pc_saft_nonassoc::pcsaft_eos::allocate_and_init (
             class(pcsaft_eos), intent(inout) eos,
             integer, intent(in) nc,
             character(len=*), intent(in) eos_label )
```

Parameters

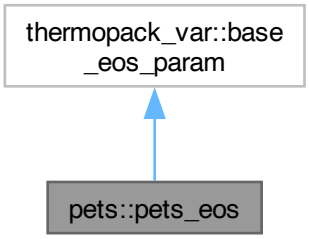
in	*nc*	Number of components
in	*eos_label*	EOS label

The documentation for this type was generated from the following file:

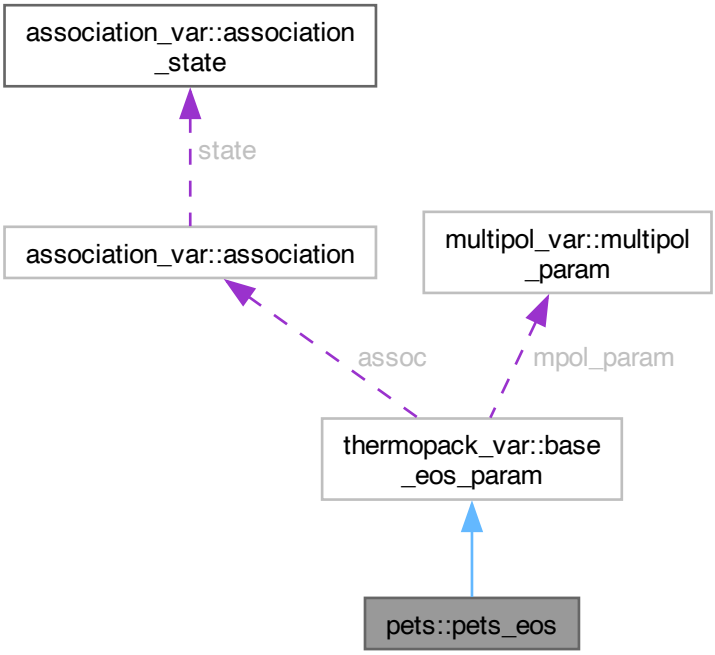
- pc_saft_nonassoc.f90

6.106 pets::pets_eos Type Reference

Inheritance diagram for pets::pets_eos:



Collaboration diagram for pets::pets_eos:



Public Member Functions

- procedure, public f_pets_tvn (eos, t, v, n, f, f_t, f_v, f_n, f_tt, f_tv, f_tn, f_vv, f_vn, f_nn)

  Gives the contribution to the reduced, residual Helmholtz function F [mol] coming from PETS' hard-sphere and dispersion contributions. All variables are in base SI units. F is defined by F(T,V,n) = sumn∗alpha_PC(rho,T,n) = sumn∗alpha_PC(sumn/V,T,n)
- procedure, public alpha_disp (eos, rho, t, n, alp, alp_rho, alp_t, alp_n, alp_rhorho, alp_rhot, alp_rhon, alp_tt, alp_tn, alp_nn)

The reduced, molar Helmholtz energy contribution from dispersion.

- procedure, public alpha_disp_tvn (eos, v, t, n, alp, alp_v, alp_t, alp_n, alp_vv, alp_vt, alp_vn, alp_tt, alp_tn, alp_nn)

  The reduced, molar Helmholtz energy contribution from dispersion.
- procedure, public alpha_pets_hs (eos, rho, t, n, alp, alp_rho, alp_t, alp_n, alp_rhorho, alp_rhot, alp_rhon, alp_tt, alp_tn, alp_nn)
- procedure, public alpha_hs_tvn (eos, v, t, n, alp, alp_v, alp_t, alp_n, alp_vv, alp_vt, alp_vn, alp_tt, alp_tn, alp_nn)

  The reduced, molar Helmholtz energy contribution from hard-sphere.
- procedure, public alpha_pets (eos, rho, t, n, alp, alp_rho, alp_t, alp_n, alp_rhorho, alp_rhot, alp_rhon, alp_tt, alp_tn, alp_nn)

  alpha_PETS = alp^{hard_sphere} + alpha^{dispersion}
- procedure, public calc_d_pets (eos, t, d, d_t, d_tt)
- procedure, public calc_potential_pets (eos, n, r, pot)
- procedure, public allocate_and_init (eos, nc, eos_label)
- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from thermopack_var::base_eos_param

- procedure(allocate_and_init_intf), deferred, public allocate_and_init (eos, nc, eos_label)
- procedure, public **dealloc** (eos)
- procedure(assign_intf), deferred, pass, public **assign_eos** (this, other)
- generic, public **assignment** assign_eos
- procedure, public **assign_base_eos_param** (this, other)

Public Attributes

- real **sigma_pets**

  [m]
- real **epsdivk_pets**

  [K]

Public Attributes inherited from thermopack_var::base_eos_param

- character(len=eosid_len) **eosid**

  Eos identifier.
- integer **eosidx**

  Eos group index.
- integer **subeosidx**

  Eos sub-index.
- integer **volumeshiftid** = 0

  0: No volume shift, 1:Peneloux shift
- logical **iselectrolyteeos** = .false.

  Used to enable electrolytes.
- type(association), pointer **assoc** => NULL()
- type(multipol_param), pointer **mpol_param** => NULL()

6.106.1 Member Function/Subroutine Documentation

6.106.1.1 allocate_and_init()

```
procedure, public pets::pets_eos::allocate_and_init (
             class(pets_eos), intent(inout) eos,
             integer, intent(in) nc,
             character(len=*), intent(in) eos_label )
```

Parameters

in	*nc*	Number of components
in	*eos_label*	EOS label

6.106.1.2 alpha_disp()

```
procedure, public pets::pets_eos::alpha_disp (
             class (pets_eos), intent(in) eos,
             real, intent(in) rho,
             real, intent(in) t,
             real, dimension(nce), intent(in) n,
             real, intent(out), optional alp,
             real, intent(out), optional alp_rho,
             real, intent(out), optional alp_t,
             real, dimension(nce), intent(out), optional alp_n,
             real, intent(out), optional alp_rhorho,
             real, intent(out), optional alp_rhot,
             real, dimension(nce), intent(out), optional alp_rhon,
             real, intent(out), optional alp_tt,
             real, dimension(nce), intent(out), optional alp_tn,
             real, dimension(nce,nce), intent(out), optional alp_nn )
```

The reduced, molar Helmholtz energy contribution from dispersion.

Parameters

in	*n*	[mol/m^3], [K], [mol]

6.106.1.3 alpha_disp_tvn()

```
procedure, public pets::pets_eos::alpha_disp_tvn (
             class (pets_eos), intent(in) eos,
             real, intent(in) v,
             real, intent(in) t,
             real, dimension(nce), intent(in) n,
             real, intent(out), optional alp,
             real, intent(out), optional alp_v,
             real, intent(out), optional alp_t,
             real, dimension(nce), intent(out), optional alp_n,
             real, intent(out), optional alp_vv,
             real, intent(out), optional alp_vt,
             real, dimension(nce), intent(out), optional alp_vn,
             real, intent(out), optional alp_tt,
             real, dimension(nce), intent(out), optional alp_tn,
             real, dimension(nce,nce), intent(out), optional alp_nn )
```

The reduced, molar Helmholtz energy contribution from dispersion.

Parameters

in	*n*	[mol/m^3], [K], [mol]

6.106.1.4 alpha_hs_tvn()

```
procedure, public pets::pets_eos::alpha_hs_tvn (
             class (pets_eos), intent(in) eos,
```

```
            real, intent(in) v,
            real, intent(in) t,
            real, dimension(nce), intent(in) n,
            real, intent(out), optional alp,
            real, intent(out), optional alp_v,
            real, intent(out), optional alp_t,
            real, dimension(nce), intent(out), optional alp_n,
            real, intent(out), optional alp_vv,
            real, intent(out), optional alp_vt,
            real, dimension(nce), intent(out), optional alp_vn,
            real, intent(out), optional alp_tt,
            real, dimension(nce), intent(out), optional alp_tn,
            real, dimension(nce,nce), intent(out), optional alp_nn )
```

The reduced, molar Helmholtz energy contribution from hard-sphere.

Parameters

in	*n*	[mol/m^3], [K], [mol]

6.106.1.5 alpha_pets()

```
procedure, public pets::pets_eos::alpha_pets (
            class (pets_eos), intent(in) eos,
            real, intent(in) rho,
            real, intent(in) t,
            real, dimension(nce), intent(in) n,
            real, intent(out), optional alp,
            real, intent(out), optional alp_rho,
            real, intent(out), optional alp_t,
            real, dimension(nce), intent(out), optional alp_n,
            real, intent(out), optional alp_rhorho,
            real, intent(out), optional alp_rhot,
            real, dimension(nce), intent(out), optional alp_rhon,
            real, intent(out), optional alp_tt,
            real, dimension(nce), intent(out), optional alp_tn,
            real, dimension(nce,nce), intent(out), optional alp_nn )
```

alpha_PETS = alp^{hard_sphere} + alpha^{dispersion}

Parameters

in	*n*	[mol/m^3], [K], [mol]
out	*alp*	[-]

6.106.1.6 alpha_pets_hs()

```
procedure, public pets::pets_eos::alpha_pets_hs (
            class (pets_eos), intent(in) eos,
            real, intent(in) rho,
            real, intent(in) t,
            real, dimension(nce), intent(in) n,
            real, intent(out) alp,
            real, intent(out), optional alp_rho,
            real, intent(out), optional alp_t,
            real, dimension(nce), intent(out), optional alp_n,
            real, intent(out), optional alp_rhorho,
            real, intent(out), optional alp_rhot,
```

```
            real, dimension(nce), intent(out), optional alp_rhon,
            real, intent(out), optional alp_tt,
            real, dimension(nce), intent(out), optional alp_tn,
            real, dimension(nce,nce), intent(out), optional alp_nn )
```

Parameters

in	*n*	[mol/m^3], [K], [mol]
out	*alp*	[-]

6.106.1.7 calc_d_pets()

```
procedure, public pets::pets_eos::calc_d_pets (
            class (pets_eos), intent(in) eos,
            real, intent(in) t,
            real, dimension(nce), intent(out) d,
            real, dimension(nce), intent(out), optional d_t,
            real, dimension(nce), intent(out), optional d_tt )
```

Parameters

in	*t*	[mol/m^3], [K], [mol]
out	*d*	[m]

6.106.1.8 calc_potential_pets()

```
procedure, public pets::pets_eos::calc_potential_pets (
            class (pets_eos), intent(in) eos,
            integer, intent(in) n,
            real, dimension(:), intent(in) r,
            real, dimension(:), intent(out) pot )
```

Parameters

in	*r*	[m]
out	*pot*	[K]

6.106.1.9 f_pets_tvn()

```
procedure, public pets::pets_eos::f_pets_tvn (
            class (pets_eos), intent(in) eos,
            real, intent(in) t,
            real, intent(in) v,
            real, dimension(nce), intent(in) n,
            real, intent(out), optional f,
            real, intent(out), optional f_t,
            real, intent(out), optional f_v,
            real, dimension(nce), intent(out), optional f_n,
            real, intent(out), optional f_tt,
            real, intent(out), optional f_tv,
            real, dimension(nce), intent(out), optional f_tn,
            real, intent(out), optional f_vv,
            real, dimension(nce), intent(out), optional f_vn,
            real, dimension(nce,nce), intent(out), optional f_nn )
```

Gives the contribution to the reduced, residual Helmholtz function F [mol] coming from PETS' hard-sphere and dispersion contributions. All variables are in base SI units. F is defined by F(T,V,n) = sumn∗alpha_PC(rho,T,n) = sumn∗alpha_PC(sumn/V,T,n)

Parameters

out	*f*	[mol]

The documentation for this type was generated from the following file:

- pets.f90

6.107 hyperdual_mod::real Interface Reference

Public Member Functions

- elemental real(dp) function **realhyperdual** (v1)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.108 utilities::safe_exp Interface Reference

Exponential function which will return "huge" instead of overflowing. Generic interface for both reals and arrays of reals.

Public Member Functions

- real function safe_exp_real (x)

  Exponential function which will return "huge" instead of overflowing. Version for reals. Called through the generic interface safe_exp.
- real function, dimension(size(x)) safe_exp_array (x)

  Exponential function which will return "huge" instead of overflowing. Version for arrays of reals. Called through the generic interface safe_exp.

6.108.1 Detailed Description

Exponential function which will return "huge" instead of overflowing. Generic interface for both reals and arrays of reals.

Author

EA, 2013-07-19

6.108.2 Member Function/Subroutine Documentation

6.108.2.1 safe_exp_array()

```
real function, dimension(size(x)) utilities::safe_exp::safe_exp_array (
            real, dimension(:), intent(in) x )
```

Exponential function which will return "huge" instead of overflowing. Version for arrays of reals. Called through the generic interface safe_exp.

Author

EA, 2013-07-19

6.108.2.2 safe_exp_real()

```
real function utilities::safe_exp::safe_exp_real (
            real, intent(in) x )
```

Exponential function which will return "huge" instead of overflowing. Version for reals. Called through the generic interface safe_exp.

Author

EA, 2013-07-19

The documentation for this interface was generated from the following file:

- utilities.f90

6.109 saftvrmie_containers::saftvrmie_aij Type Reference

Container for a_ij and differentials.

Public Member Functions

- procedure, public **mirror** (this)
- procedure, public **assign_saftvrmie_aij** (this, other)
- generic, public **assignment** (this, other)

Public Attributes

- real, dimension(:,:), allocatable **am**
- real, dimension(:,:), allocatable **am_t**
- real, dimension(:,:), allocatable **am_v**
- real, dimension(:,:), allocatable **am_tt**
- real, dimension(:,:), allocatable **am_vv**
- real, dimension(:,:), allocatable **am_tv**
- real, dimension(:,:), allocatable **am_vvv**
- real, dimension(:,:), allocatable **am_vvt**
- real, dimension(:,:), allocatable **am_vtt**
- real, dimension(:,:,:), allocatable **am_n**
- real, dimension(:,:,:), allocatable **am_tn**
- real, dimension(:,:,:), allocatable **am_vn**
- real, dimension(:,:,:), allocatable **am_vvn**
- real, dimension(:,:,:), allocatable **am_vtn**
- real, dimension(:,:,:,:), allocatable **am_nn**
- real, dimension(:,:,:,:), allocatable **am_vnn**

6.109.1 Detailed Description

Container for a_ij and differentials.
The documentation for this type was generated from the following file:

- saftvrmie_containers.f90

6.110 saftvrmie_datadb::saftvrmie_data Type Reference

PURE COMPONENT PARAMETERS. This data structure stores pure component parameters for the SAFT-VRQ Mie EoS.

Public Attributes

- integer **eosidx**
- character(len=uid_len) **compname**
- real **m**

  [-]. Mean number of segments.
- real **sigma**

  [m]. Temperature-independent segment diameter.
- real **eps_depth_divk**

  [K]. Well depth divided by Boltzmann's c.
- real **lambda_a**

  [] attractive exponent of the Mie potential
- real **lambda_r**

  [] repulsive exponent of the Mie potential
- real **mass**

  Segment mass, i.e. molecule mass dividided by number of segments [kg].
- real **eps**

  [J/mol].
- real **beta**

  [-]. Also known as kappa in SAFT literature.
- integer **assoc_scheme**
- integer **fh_order**
- character(len=bibref_len) **bib_ref**
- character(len=ref_len) **ref**

6.110.1 Detailed Description

PURE COMPONENT PARAMETERS. This data structure stores pure component parameters for the SAFT-VRQ Mie EoS.
The documentation for this type was generated from the following file:

- saftvrmie_datadb.f90

6.111 saftvrmie_containers::saftvrmie_dhs Type Reference

Container for hard-sphere diameter and differentials Also used for the Feynman–Hibbs D variable.

Public Member Functions

- procedure, public **update_symmetric_diameters** (this)
- procedure, public **assign_saftvrmie_dhs** (this, other)
- generic, public **assignment** (this, other)

Public Attributes

- real, dimension(:,:), allocatable **d**

  Hard sphere diameter.
- real, dimension(:,:), allocatable **d_t**

  Temperature differential of hard sphere diameter.
- real, dimension(:,:), allocatable **d_tt**

  Second temperature differential of hard sphere diameter.

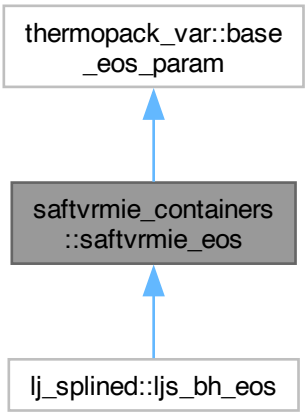
6.111.1 Detailed Description

Container for hard-sphere diameter and differentials Also used for the Feynman–Hibbs D variable.
The documentation for this type was generated from the following file:

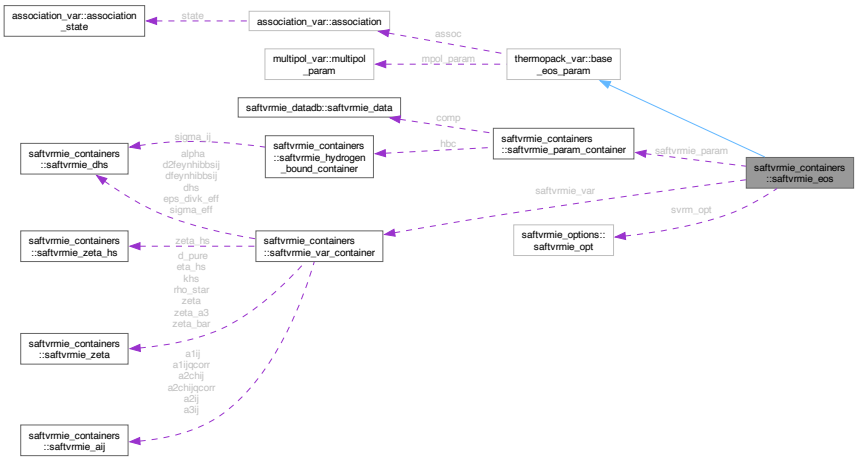
- saftvrmie_containers.f90

6.112 saftvrmie_containers::saftvrmie_eos Type Reference

Inheritance diagram for saftvrmie_containers::saftvrmie_eos:



Collaboration diagram for saftvrmie_containers::saftvrmie_eos:



Public Member Functions

- procedure, public **dealloc** (eos)
- procedure, public allocate_and_init (eos, nc, eos_label)
- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from thermopack_var::base_eos_param

- procedure(allocate_and_init_intf), deferred, public allocate_and_init (eos, nc, eos_label)
- procedure, public **dealloc** (eos)
- procedure(assign_intf), deferred, pass, public **assign_eos** (this, other)
- generic, public **assignment** assign_eos
- procedure, public **assign_base_eos_param** (this, other)

Public Attributes

- logical **ovner_of_saftvrmie_param** = .false.
- type(saftvrmie_param_container), pointer **saftvrmie_param** => NULL()
- type(saftvrmie_var_container), pointer **saftvrmie_var** => NULL()
- logical **ovner_of_svrm_opt** = .false.
- type(saftvrmie_opt), pointer **svrm_opt** => NULL()

Public Attributes inherited from thermopack_var::base_eos_param

- character(len=eosid_len) **eosid**

  Eos identifier.
- integer **eosidx**

  Eos group index.
- integer **subeosidx**

  Eos sub-index.
- integer **volumeshiftid** = 0

  0: No volume shift, 1:Peneloux shift
- logical **iselectrolyteeos** = .false.

  Used to enable electrolytes.
- type(association), pointer **assoc** => NULL()
- type(multipol_param), pointer **mpol_param** => NULL()

6.112.1 Member Function/Subroutine Documentation

6.112.1.1 allocate_and_init()

```
procedure, public saftvrmie_containers::saftvrmie_eos::allocate_and_init (
              class(saftvrmie_eos), intent(inout) eos,
              integer, intent(in) nc,
              character(len=*), intent(in) eos_label )
```

Parameters

in	*eos_label*	EOS label

The documentation for this type was generated from the following file:

- saftvrmie_containers.f90

6.113 saftvrmie_options::saftvrmie_opt Type Reference

Public Member Functions

- procedure, public saftvrmieaij_model_options (svrm_o, model, hs_reference)

  Model options....
- procedure, public set_r_cut (svrm_o, r_c)

  Set r_cut, and enable truncation correction.

- procedure, public truncation_correction_model_control (svrm_o, truncation, shift)

  Enable/disable truncation and shift correction.
- procedure, public check_model_consitency (svrm_o)

  Check that model parameters are consistent.
- procedure, public **set_lafitte_option** (svrm_o)
- procedure, public **set_hs_reference** (svrm_o, hs_reference)
- procedure, public **test_fmt_compatibility** (this, is_fmt_consistent, na_enabled)

  Test if SAFT-VR Mie model setup is comaptible with the Fundamental Measure Theory (FMT)
- procedure, public **print** (this)
- procedure, public **assign_saftvrmie_model_options** (this, other)
- generic, public **assignment** (this, other)

Public Attributes

- integer **quantum_correction** = 0
- integer **quantum_correction_hs** = 0
- integer **quantum_correction_spec** = 0
- logical **quantum_correct_a2** = .true.
- logical **use_epsrule_lafitte** = .true.
- logical **exact_binary_dhs** =.false.
- logical **exact_crosspot_eff** =.true.
- integer **zeta_mixing_rule** = ZETA_LAFITTE
- integer **hardsphere_eos** = HS_EOS_ORIGINAL
- integer **khs_eos** = KHS_EOS_LAFITTE
- integer **pure_hs_eos** = PURE_HS_CS
- logical **enable_hs_extra** = .false.
- logical **option**
- logical **to**
- logical **enable**
- logical **disable**
- logical **extra**
- logical **term**
- logical **hard**
- logical **sphere**
- logical **reference**
- integer **a3_model** = A3_LAFITTE
- logical **enable_hs** = .true.
- logical **contribution**
- logical **enable_a1** = .true.
- logical **a1**
- logical **enable_a3** = .true.
- logical **a3**
- logical **enable_a2** = .true.
- logical **a2**
- logical **enable_chain** = .true.
- logical **chain**
- logical **enable_truncation_correction** = .false.
- logical **truncation**
- logical **correction**
- logical **enable_shift_correction** = .false.
- logical **shift**
- real **r_cut** = 3.5
- real **truncation**
- real **radius**
- logical **use_temp_cache** = .false.

6.113.1 Member Function/Subroutine Documentation

6.113.1.1 check_model_consitency()

```
procedure, public saftvrmie_options::saftvrmie_opt::check_model_consitency (
             class(saftvrmie_opt), intent(in) svrm_o )
```

Check that model parameters are consistent.

Author

Morten Hammer, March 2019

6.113.1.2 saftvrmieaij_model_options()

```
procedure, public saftvrmie_options::saftvrmie_opt::saftvrmieaij_model_options (
             class(saftvrmie_opt), intent(inout) svrm_o,
             integer, intent(in) model,
             integer, intent(in), optional hs_reference )
```

Model options....

Author

Morten Hammer, January 2019

6.113.1.3 set_r_cut()

```
procedure, public saftvrmie_options::saftvrmie_opt::set_r_cut (
             class(saftvrmie_opt), intent(inout) svrm_o,
             real, intent(in) r_c )
```

Set r_cut, and enable truncation correction.

Author

Morten Hammer, November 2018

6.113.1.4 truncation_correction_model_control()

```
procedure, public saftvrmie_options::saftvrmie_opt::truncation_correction_model_control (
             class(saftvrmie_opt), intent(inout) svrm_o,
             logical, intent(in) truncation,
             logical, intent(in) shift )
```

Enable/disable truncation and shift correction.

Author

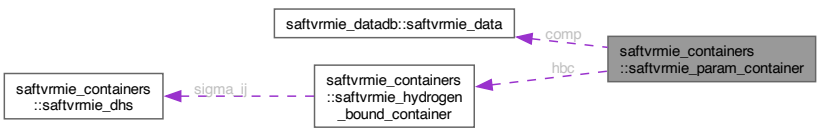
Morten Hammer, November 2018

The documentation for this type was generated from the following file:

- saftvrmie_options.f90

6.114 saftvrmie_containers::saftvrmie_param_container Type Reference

Container for SAFT-VR Mie static parameters.
Collaboration diagram for saftvrmie_containers::saftvrmie_param_container:



Public Member Functions

- procedure, public **print** (this)
- procedure, public **assign_saftvrmie_param_container** (this, other)
- generic, public **assignment** (this, other)

Public Attributes

- type(saftvrmie_data), dimension(:), allocatable **comp**

  Component parameters.
- real, dimension(:,:), allocatable **kij**

  Binary interaction parameters for the well depth.
- real, dimension(:,:), allocatable **gamma_ij**

  Binary interaction parameters for the repulsive exponent.
- real, dimension(:,:), allocatable **lij**

  Binary interaction parameters for sigma.
- real, dimension(:,:), allocatable **alpha_ij**

  van der Waals-like attractive constant
- real, dimension(:,:,:), allocatable **f_alpha_ij**

  Function of alpha.
- real, dimension(:,:), allocatable **lambda_a_ij**

  Binary attractive exponent of the Mie potential.
- real, dimension(:,:), allocatable **lambda_r_ij**

  Binary repulsive exponent of the Mie potential.
- real, dimension(:,:), allocatable **sigma_ij**

  Temperature-independent segment diameter (m)
- real, dimension(:,:), allocatable **eps_divk_ij**

  Binary well depth divided by Boltzmann's k (K)
- real, dimension(:,:), allocatable **cij**

  Binary Mie C factor.
- real, dimension(:,:), allocatable **dfeynhibbsparam_ij**

  T-independent part of binary Feynman–Hibbs D parameter.
- real, dimension(:), allocatable **ms**

  Copy of comp(:)ms for easy looping.
- real, dimension(:,:), allocatable **sigma_ij_cube**

  Cube of temperature-independent segment diameter (m3)
- real, dimension(:,:), allocatable **quantum_const_1a_ij**

  Parameters in the quantum correction, first order - attractive.
- real, dimension(:,:), allocatable **quantum_const_1r_ij**

  Parameters in the quantum correction, first order - repulsive.
- real, dimension(:,:), allocatable **quantum_const_2a_ij**

  Parameters in the quantum correction, second order - attractive.
- real, dimension(:,:), allocatable **quantum_const_2r_ij**

  Parameters in the quantum correction, second order - repulsive.
- logical **isselfassociating** = .false.
- type(saftvrmie_hydrogen_bound_container) **hbc**

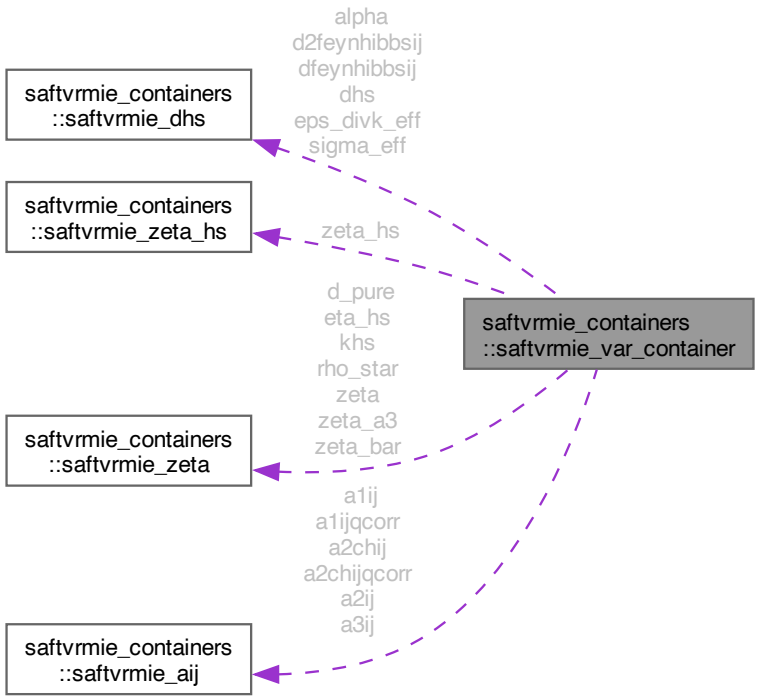
6.114.1 Detailed Description

Container for SAFT-VR Mie static parameters.
The documentation for this type was generated from the following file:

- saftvrmie_containers.f90

6.115 saftvrmie_containers::saftvrmie_var_container Type Reference

Container for SAFT-VR Mie common variables To be claculated only once per state.
Collaboration diagram for saftvrmie_containers::saftvrmie_var_container:



Public Member Functions

- procedure, public **assign_saftvrmie_var_container** (this, other)
- generic, public **assignment** (this, other)

Public Attributes

- real **temperature_dhs** = -1.0

  Temperature of last update.
- real **temperature_eps** = -1.0
- type(saftvrmie_dhs) **dhs**

  Hard sphere diameter.
- type(saftvrmie_dhs) **sigma_eff**

  Effective sigma for the quantum corrected potential.
- type(saftvrmie_dhs) **eps_divk_eff**

  Effective epsilon for the quantum corrected potential.
- type(saftvrmie_dhs) **dfeynhibbsij**

  Feynman–Hibbs D variable.
- type(saftvrmie_dhs) **d2feynhibbsij**

  Feynman–Hibbs D variable squared.
- type(saftvrmie_dhs) **alpha**

  Dimensionless van der Waals energy.

- type(saftvrmie_zeta_hs) **zeta_hs**

  Moments of the number density for hs-module.
- type(saftvrmie_zeta) **zeta**

  Hypotetical pure fluid packing fraction.
- type(saftvrmie_zeta) **zeta_bar**

  Packing fraction.
- type(saftvrmie_zeta) **zeta_a3**

  Zeta used as prefactor in a3.
- type(saftvrmie_zeta) **khs**

  Isothermal hard sphere compressibillity factor.
- type(saftvrmie_zeta) **eta_hs**

  Hard sphere packing fraction.
- type(saftvrmie_zeta) **d_pure**

  Pure fluid reference HS diameter.
- type(saftvrmie_zeta) **rho_star**

  Rho star.
- type(saftvrmie_aij) **a1ij**

  A1_ij.
- type(saftvrmie_aij) **a1ijqcorr**

  Additive quantum corrections to A1_ij.
- type(saftvrmie_aij) **a2chij**

  (A2/(1-chi))_ij
- type(saftvrmie_aij) **a2chijqcorr**

  Additive quantum corrections to A2chij.
- type(saftvrmie_aij) **a2ij**

  A2_ij.
- type(saftvrmie_aij) **a3ij**

  A3_ij.

6.115.1 Detailed Description

Container for SAFT-VR Mie common variables To be claculated only once per state.
The documentation for this type was generated from the following file:

- saftvrmie_containers.f90

6.116 saftvrmie_containers::saftvrmie_zeta Type Reference

Container for zeta and differentials (also used for functions of zeta)

Public Member Functions

- procedure, public **assign_saftvrmie_zeta** (this, other)
- generic, public **assignment** (this, other)

Public Attributes

- real **zx**

  Hypotetical pure fluid packing fraction.
- real **zx_t**

  Temperature differential of hypotetical pure fluid packing fraction.
- real **zx_tt**

  Second temperature differential of hypotetical pure fluid packing fraction.

- real **zx_v**

  Volume differential of hypotetical pure fluid packing fraction.
- real **zx_vv**

  Second volume differential of hypotetical pure fluid packing fraction.
- real **zx_tv**

  Temperature and volume differential of hypotetical pure fluid packing fraction.
- real, dimension(:), allocatable **zx_n**

  Mol number differential of hypotetical pure fluid packing fraction.
- real, dimension(:), allocatable **zx_vn**

  Mol number and volume differential of hypotetical pure fluid packing fraction.
- real, dimension(:), allocatable **zx_tn**

  Mol number and temperature differential of hypotetical pure fluid packing fraction.
- real, dimension(:,:), allocatable **zx_nn**

  Second mol number differential of hypotetical pure fluid packing fraction.
- real **zx_vvv**

  Three time volume differential of hypotetical pure fluid packing fraction.
- real **zx_vvt**

  Temperature and twice volume differential of hypotetical pure fluid packing fraction.
- real **zx_vtt**

  Twice temperature and volume differential of hypotetical pure fluid packing fraction.
- real, dimension(:), allocatable **zx_vtn**

  Mol number, temperature and volume differential of hypotetical pure fluid packing fraction.
- real, dimension(:), allocatable **zx_vvn**

  Mol number and twice volume and volume differential of hypotetical pure fluid packing fraction.
- real, dimension(:,:), allocatable **zx_vnn**

  Volume and twice mol number and temperature differential of hypotetical pure fluid packing fraction.

6.116.1 Detailed Description

Container for zeta and differentials (also used for functions of zeta)
The documentation for this type was generated from the following file:

- saftvrmie_containers.f90

6.117 saftvrmie_containers::saftvrmie_zeta_hs Type Reference

Container for mu and zeta's (2 and 3). These are moments of the number density (2,3) and mu (1)

Public Member Functions

- procedure, public **assign_saftvrmie_zeta_hs** (this, other)
- generic, public **assignment** (this, other)

Public Attributes

- real, dimension(3) **zet**

  Moments of the number density and mu.
- real, dimension(3) **zet_t**

  Temperature differential of the moments of the number density and mu.
- real, dimension(3) **zet_tt**

  Second temperature differential of the moments of the number density and mu.
- real, dimension(3) **zet_v**

  Volume differential of the moments of the number density and mu.

- real, dimension(3) **zet_vv**

  Second volume differential of the moments of the number density and mu.
- real, dimension(3) **zet_tv**

  Temperature and volume differential of the moments of the number density and mu.
- real, dimension(:,:), allocatable **zet_n**

  Mol number differential of the moments of the number density and mu.
- real, dimension(:,:), allocatable **zet_vn**

  Mol number and volume differential of the moments of the number density and mu.
- real, dimension(:,:), allocatable **zet_tn**

  Mol number and temperature differential of the moments of the number density and mu.

6.117.1 Detailed Description

Container for mu and zeta's (2 and 3). These are moments of the number density (2,3) and mu (1)
The documentation for this type was generated from the following file:

- saftvrmie_containers.f90

6.118 saturated_densities::sat_densities Type Reference

Class calculating saturated densities.

Public Member Functions

- procedure, public **init** (this, comp_name, phase)
- procedure, public **density** (this, t)

Public Attributes

- character(len=40), public **compname**
- integer **correlation_id** = 0
- real, public **t_reducing**
- real, public **rho_reducing**
- integer **n_param** = 0
- real, dimension(6) **n_sat**
- real, dimension(6) **expo_sat**

6.118.1 Detailed Description

Class calculating saturated densities.
The documentation for this type was generated from the following file:

- saturated_densities.f90

6.119 multiparameter_base::satdeltaestimate_intf Interface Reference

Public Member Functions

- real function satdeltaestimate_intf (this, tau, phase)

6.119.1 Constructor & Destructor Documentation

6.119.1.1 satdeltaestimate_intf()

```
real function multiparameter_base::satdeltaestimate_intf::satdeltaestimate_intf (
             class(meos) this,
             real, intent(in) tau,
             integer, intent(in) phase )  [virtual]
```

Parameters

in	*tau*	Reduced temperature (-)
in	*phase*	Phase flag

Returns

Reduced density (-)

The documentation for this interface was generated from the following file:

- multiparameter_base.f90

6.120 satdensdatadb::satdensdata Type Reference

Public Attributes

- character(len=8) **ident**

  The component ID.
- character(len=40) **name**

  The component name.
- character(len=1) **phase**

  Phase (G/L)
- real **tr**

  Reducing temperature (K)
- real **rhor**

  Reducing density (mol/l)
- integer **correlation_id**
- integer **n_param**
- real, dimension(6) **n**
- real, dimension(6) **t**

The documentation for this type was generated from the following file:

- satdensdatadb.f90

6.121 extcsp::shape_diff Type Reference

Derivatives. Uses the notation on pages 115-120 in Michelsen & Mollerup.

Public Member Functions

- procedure, public shape_diff_alloc (sdiff, nce)

  Allocate differential struct.
- procedure, public shape_diff_dealloc (sdiff)

  Deallocate differential struct.

Public Attributes

- real **h**
- real **f**
- real **ht**
- real **ft**
- real **htt**
- real **ftt**
- real, dimension(:), allocatable **hi**

- real, dimension(:), allocatable **hit**
- real, dimension(:), allocatable **fi**
- real, dimension(:), allocatable **fit**
- real, dimension(:,:), allocatable **hij**
- real, dimension(:,:), allocatable **fij**
- real **v0v**
- real **v0vv**
- real **t0t**
- real **t0tt**
- real, dimension(:), allocatable **t0ti**
- real, dimension(:), allocatable **t0i**
- real, dimension(:), allocatable **v0vi**
- real, dimension(:), allocatable **v0i**
- real, dimension(:,:), allocatable **v0ij**
- real, dimension(:,:), allocatable **t0ij**
- real **d**
- real **b**
- real **bt**
- real **dt**
- real **btt**
- real **dtt**
- real, dimension(:), allocatable **bi**
- real, dimension(:), allocatable **bit**
- real, dimension(:), allocatable **di**
- real, dimension(:), allocatable **dit**
- real, dimension(:,:), allocatable **bij**
- real, dimension(:,:), allocatable **dij**
- real **m**
- real **mt0**
- real **mt0t0**
- real **mt0v0**
- real **mv0**
- real **mv0v0**
- real **ff**
- real **ff_t**
- real **ff_v**
- real **ff_tt**
- real **ff_vv**
- real **ff_tv**
- real, dimension(:), allocatable **ff_i**
- real, dimension(:), allocatable **ff_ti**
- real, dimension(:), allocatable **ff_vi**
- real, dimension(:,:), allocatable **ff_ij**
- logical **v0i_set** = .false.

6.121.1 Detailed Description

Derivatives. Uses the notation on pages 115-120 in Michelsen & Mollerup.

6.121.2 Member Function/Subroutine Documentation

6.121.2.1 shape_diff_alloc()

```
procedure, public extcsp::shape_diff::shape_diff_alloc (
             class(shape_diff), intent(inout) sdiff,
             integer, intent(in) nce )
```

Allocate differential struct.

Author

MH, 2013-11-27

6.121.2.2 shape_diff_dealloc()

```
procedure, public extcsp::shape_diff::shape_diff_dealloc (
             class(shape_diff), intent(inout) sdiff )
```

Deallocate differential struct.

Author

MH, 2013-11-27

The documentation for this type was generated from the following file:

- extcsp.f90

6.122 hyperdual_mod::sign Interface Reference

Public Member Functions

- elemental type(hyperdual) function **sign_dd** (v1, v2)
- elemental type(hyperdual) function **sign_dr** (v1, v2)
- elemental type(hyperdual) function **sign_rd** (v1, v2)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.123 hyperdual_mod::sin Interface Reference

Public Member Functions

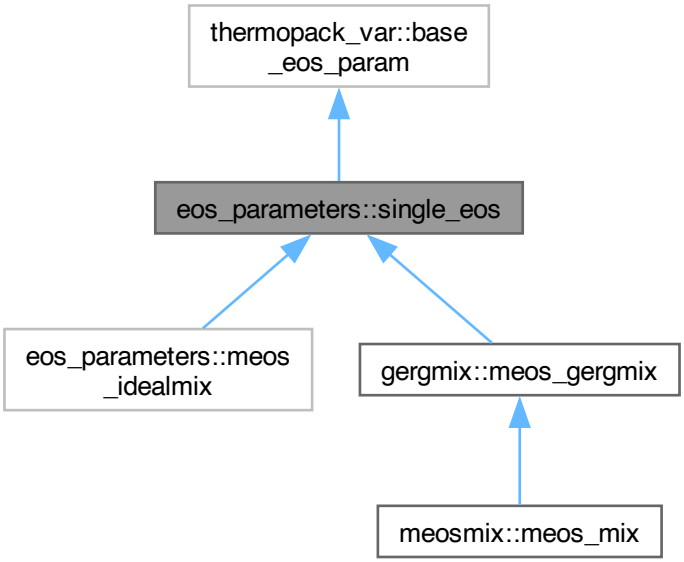
- elemental type(hyperdual) function **sinhyperdual** (v1)

The documentation for this interface was generated from the following file:

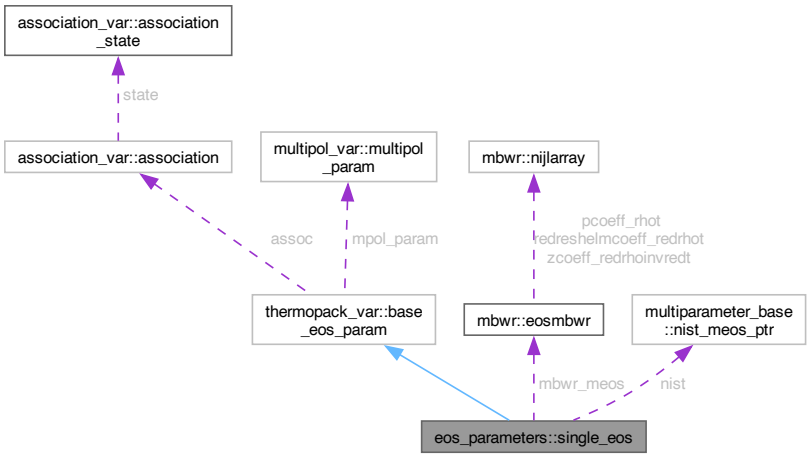
- hyperdual_mod.f90

6.124 eos_parameters::single_eos Type Reference

Inheritance diagram for eos_parameters::single_eos:



Collaboration diagram for eos_parameters::single_eos:



Public Member Functions

- procedure, public **dealloc** (eos)
- procedure, public allocate_and_init (eos, nc, eos_label)
- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from thermopack_var::base_eos_param

- procedure(allocate_and_init_intf), deferred, public allocate_and_init (eos, nc, eos_label)
- procedure, public **dealloc** (eos)
- procedure(assign_intf), deferred, pass, public **assign_eos** (this, other)
- generic, public **assignment** assign_eos
- procedure, public **assign_base_eos_param** (this, other)

Public Attributes

- type(eosmbwr), dimension(:), allocatable **mbwr_meos**
- type(nist_meos_ptr), dimension(:), allocatable **nist**

Public Attributes inherited from thermopack_var::base_eos_param

- character(len=eosid_len) **eosid**

  Eos identifier.
- integer **eosidx**

  Eos group index.
- integer **subeosidx**

  Eos sub-index.
- integer **volumeshiftid** = 0

  0: No volume shift, 1:Peneloux shift
- logical **iselectrolyteeos** = .false.

  Used to enable electrolytes.
- type(association), pointer **assoc** => NULL()
- type(multipol_param), pointer **mpol_param** => NULL()

6.124.1 Member Function/Subroutine Documentation

6.124.1.1 allocate_and_init()

```
procedure, public eos_parameters::single_eos::allocate_and_init (
              class(single_eos), intent(inout) eos,
              integer, intent(in) nc,
              character(len=*), intent(in) eos_label )
```

Parameters

in	*nc*	Number of components
in	*eos_label*	EOS label

The documentation for this type was generated from the following file:

- eos_parameters.f90

6.125 cubic_eos::singledata Type Reference

Public Attributes

- real **omegaa**

  Cubic EoS quantities for individual components.
- real **omegab**
- real **omegac**
- real **zcrit**
- real **a**

Energy constant in front of alpha. Units: Pa∗L^2/mol^2.

- real **b**

  Covolume parameter. Units: L/mol.
- real **c**

  Parameter in Patel-Teja EoS. Units: L/mol.
- real **acf**

  Acentric factor to use for the cubic EoS [-].
- real **tc**

  Critical temperature to use in the cubic EoS [K].
- real **pc**

  Critical pressure to use in the cubic EoS [Pa].
- integer **alphamethod** =-1
- character(len=short_label_len) **alphacorrname**
- real **alpha**
- real **dalphadt**
- real **d2alphadt2**

  alpha is dimensionless
- real, dimension(3) **alphaparams** =-1e10
- integer **betamethod** =1
- character(len=12) **betacorrname** = "Classic "
- real **beta**
- real **dbetadt**
- real **d2betadt2**

  beta is dimensionless
- real, dimension(3) **betaparams** =-1e10

The documentation for this type was generated from the following file:

- cubic_eos.f90

6.126 hyperdual_mod::sinh Interface Reference

Public Member Functions

- elemental type(hyperdual) function **sinhhyperdual** (v1)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.127 solid_correlation_datadb::solid_correlation_data Type Reference

This data structure stores parameters for sublimation and melting line correlations.

Public Attributes

- character(len=uid_len) **compname**
- character(len=4) **correlation**

  Correlations type.
- real **triple_temperature**

  [K]. Triple point temperature.
- real **minimum_temperature**

  [K]. Minimum temperature for sublimation line.
- real **maximum_temperature**

[K]. Maximum temperature for melting line.

- real **reducing_pressure**

  [Pa]. Pressure scaling parameter.

- real **reducing_temperature**

  [K]. Temperature reducing parameter.

- integer **n_coeff**

  Number of coefficients.

- integer **n_coeff_1**

  Number of coefficients for type one terms.

- integer **n_coeff_2**

  Number of coefficients for type two terms.

- integer **n_coeff_3**

  Number of coefficients for type three terms.

- real, dimension(6) **coeff**

  Correlation coefficients.

- real, dimension(6) **exponents**

  Correlation exponents.

- character(len=bibref_len) **bib_ref**

  Bibliograpic reference.

- character(len=ref_len) **ref**

  Parameter set.

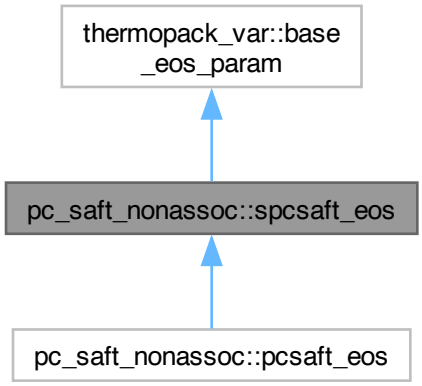
6.127.1 Detailed Description

This data structure stores parameters for sublimation and melting line correlations.
The documentation for this type was generated from the following file:

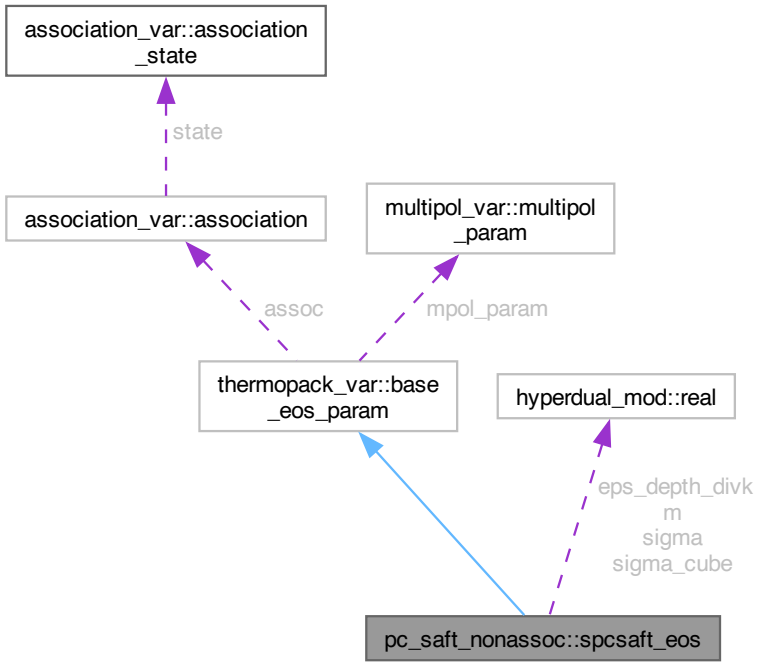
- solid_correlation_datadb.f90

6.128 pc_saft_nonassoc::spcsaft_eos Type Reference

Inheritance diagram for pc_saft_nonassoc::spcsaft_eos:



Collaboration diagram for pc_saft_nonassoc::spcsaft_eos:



Public Member Functions

- procedure, public **dealloc** (eos)
- procedure, public allocate_and_init (eos, nc, eos_label)
- procedure, pass, public **assign_eos** (this, other)

Public Member Functions inherited from thermopack_var::base_eos_param

- procedure(allocate_and_init_intf), deferred, public allocate_and_init (eos, nc, eos_label)
- procedure, public **dealloc** (eos)
- procedure(assign_intf), deferred, pass, public **assign_eos** (this, other)
- generic, public **assignment** assign_eos
- procedure, public **assign_base_eos_param** (this, other)

Public Attributes

- real, dimension(:), allocatable **m**

  [-]
- real, dimension(:,:), allocatable **sigma**

  [m]
- real, dimension(:,:), allocatable **eps_depth_divk**

  [K]
- real, dimension(:,:), allocatable **sigma_cube**

  [m^3]

Public Attributes inherited from thermopack_var::base_eos_param

- character(len=eosid_len) **eosid**

  Eos identifier.
- integer **eosidx**

  Eos group index.
- integer **subeosidx**

  Eos sub-index.
- integer **volumeshiftid** = 0

  0: No volume shift, 1:Peneloux shift
- logical **iselectrolyteeos** = .false.

  Used to enable electrolytes.
- type(association), pointer **assoc** => NULL()
- type(multipol_param), pointer **mpol_param** => NULL()

6.128.1 Member Function/Subroutine Documentation

6.128.1.1 allocate_and_init()

```
procedure, public pc_saft_nonassoc::spcsaft_eos::allocate_and_init (
              class(spcsaft_eos), intent(inout) eos,
              integer, intent(in) nc,
              character(len=*), intent(in) eos_label )
```

Parameters

in	*nc*	Number of components
in	*eos_label*	EOS label

The documentation for this type was generated from the following file:

- pc_saft_nonassoc.f90

6.129 hyperdual_mod::sqrt Interface Reference

Public Member Functions

- elemental type(hyperdual) function **sqrthyperdual** (v1)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.130 vls::state Type Reference

Thermo state, used for debugging.

Public Member Functions

- procedure set_state (st, nd, t, p, z, beta, xx, phasevec)
- procedure get_state (st, nd, t, p, z, beta, xx, phasevec)
- procedure get_state_no_z (st, nd, t, p, beta, xx, phasevec)
- procedure **print_state** (st)

Public Attributes

- integer **nd**
- real **t**
- real **p**
- real, dimension(:), allocatable **z**
- real, dimension(:), allocatable **beta**
- real, dimension(:,:), allocatable **xx**
- integer, dimension(:), allocatable **phasevec**

6.130.1 Detailed Description

Thermo state, used for debugging.

6.130.2 Member Function/Subroutine Documentation

6.130.2.1 get_state()

```
procedure vls::state::get_state (
             class(state), intent(in) st,
             integer, intent(out) nd,
             real, intent(out) t,
             real, intent(out) p,
             real, dimension(:), intent(out) z,
             real, dimension(:), intent(out) beta,
             real, dimension(:,:), intent(out) xx,
             integer, dimension(:), intent(out) phasevec )
```

Parameters

out	*nd*	Number of stabel phases found [-]
out	*beta*	Phase molar fractions [mol/mol]
out	*xx*	Phase molar compozition [mol/mol]
out	*z*	Overall molar compozition [mol/mol]
out	*t*	Temperature [K]
out	*p*	Specified pressure [Pa]
out	*phasevec*	Phase identifier. Not to be trused [-]

6.130.2.2 get_state_no_z()

```
procedure vls::state::get_state_no_z (
             class(state), intent(in) st,
             integer, intent(out) nd,
             real, intent(out) t,
             real, intent(out) p,
             real, dimension(:), intent(out) beta,
             real, dimension(:,:), intent(out) xx,
             integer, dimension(:), intent(out) phasevec )
```

Parameters

out	*nd*	Number of stabel phases found [-]
out	*beta*	Phase molar fractions [mol/mol]
out	*xx*	Phase molar compozition [mol/mol]
out	*t*	Temperature [K]
out	*p*	Specified pressure [Pa]

Parameters

out	*phasevec*	Phase identifier. Not to be trused [-]

6.130.2.3 set_state()

```
procedure vls::state::set_state (
             class(state), intent(inout) st,
             integer, intent(in) nd,
             real, intent(in) t,
             real, intent(in) p,
             real, dimension(:), intent(in) z,
             real, dimension(:), intent(in) beta,
             real, dimension(:,:), intent(in) xx,
             integer, dimension(:), intent(in) phasevec )
```

Parameters

in	*nd*	Number of stabel phases found [-]
in	*beta*	Phase molar fractions [mol/mol]
in	*xx*	Phase molar compozition [mol/mol]
in	*z*	Overall molar compozition [mol/mol]
in	*t*	Temperature [K]
in	*p*	Specified pressure [Pa]
in	*phasevec*	Phase identifier. Not to be trused [-]

The documentation for this type was generated from the following file:

- vls.f90

6.131 hyperdual_mod::sum Interface Reference

Public Member Functions

- pure type(hyperdual) function **sumhyperdual** (v1, mask)
- pure type(hyperdual) function, dimension(:), allocatable **sumhyperdual2** (v1, dim)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.132 hyperdual_mod::tan Interface Reference

Public Member Functions

- elemental type(hyperdual) function **tanhyperdual** (v1)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.133 hyperdual_mod::tanh Interface Reference

Public Member Functions

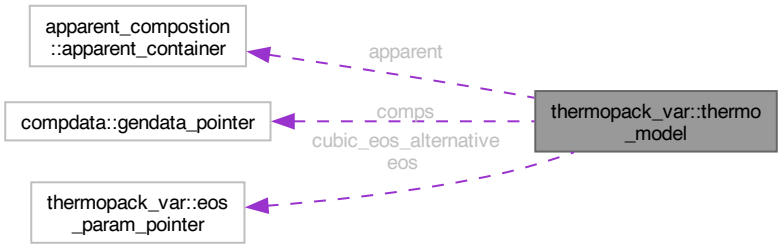
- elemental type(hyperdual) function **tanhhyperdual** (v1)

The documentation for this interface was generated from the following file:

- hyperdual_mod.f90

6.134 thermopack_var::thermo_model Type Reference

Collaboration diagram for thermopack_var::thermo_model:



Public Member Functions

- procedure, public **dealloc** (model)
- procedure, public **is_model_container** (model, index)

Public Attributes

- integer model_idx

  A complete ThermoPack model. Holds information about the EoS, the mixture, and the computational solver options.
- integer **nph** =3
- integer **nc** =0
- integer **eoslib** =0
- integer **eosidx** =0
- character(len=label_len) **label**
- integer **liq_vap_discr_method** =PSEUDO_CRIT_MOLAR_VOLUME

  Gas constant usde by current model.
- real **rgas** = Rgas_default

  J/mol/K.
- real **krgas** = 1000.0∗Rgas_default

  J/kmol/K Temperature/pressure min/max values used for solvers.
- real **tptmax** = 999.0

  K.
- real **tptmin** = 80.0

  K.
- real **tppmax** = 1.0e8

  Pa.
- real **tppmin** = 1.0e1

  Pa.
- type(apparent_container), pointer **apparent** => NULL()
- type(gendata_pointer), dimension(:), allocatable **comps**
- character(len=eosid_len), dimension(:), allocatable **complist**
- type(eos_param_pointer), dimension(:), allocatable **eos**
- logical **need_alternative_eos**
- type(eos_param_pointer), dimension(:), allocatable **cubic_eos_alternative**

6.134.1 Member Data Documentation

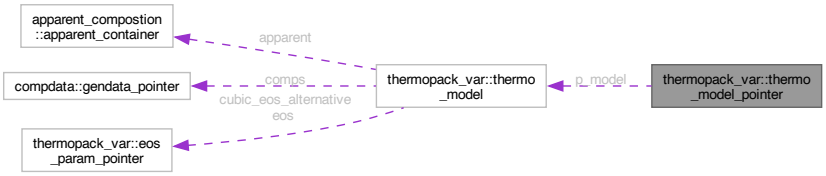
6.134.1.1 model_idx

`integer thermopack_var::thermo_model::model_idx`
A complete ThermoPack model. Holds information about the EoS, the mixture, and the computational solver options. Model is active if this equals activated_model_idx
The documentation for this type was generated from the following file:

- thermopack_var.f90

6.135 thermopack_var::thermo_model_pointer Type Reference

Collaboration diagram for thermopack_var::thermo_model_pointer:



Public Attributes

- type(thermo_model), pointer **p_model** => NULL()

  A trivial type that only contains a pointer to thermo_model. This type is needed because gfortran does not allow arrays of pointer to base_eos_param, whereas arrays of the thermo_model_pointer type is allowed.

The documentation for this type was generated from the following file:

- thermopack_var.f90

6.136 unifacdata::unifaccomp Type Reference

Public Attributes

- character(len=20) **formula**
- character(len=20) **uid**
- integer, dimension(nsubgroups) **v**

The documentation for this type was generated from the following file:

- unifacdata.f90

6.137 unifac::unifacdb Type Reference

Structure for active unifac parameters.

Public Member Functions

- procedure **dealloc** (u)
- procedure **assign_unifacdb** (u1, u2)
- generic, public **assignment** (u1, u2)

Public Attributes

- logical **floryhuggins**
- logical **stavermanguggenheim**
- integer, dimension(:), allocatable **maingroupmapping**
- integer, dimension(:,:), allocatable **vik**

  (nc,ng) [-] Number of groups in one component
- real, dimension(:,:), allocatable **ajk**

  (ng,ng) [K] Group interaction energy
- real, dimension(:,:), allocatable **bjk**

  (ng,ng) [-] Group interaction energy
- real, dimension(:,:), allocatable **cjk**

  (ng,ng) [1/K] Group interaction energy
- real, dimension(:), allocatable **qk**

  (ng) [] Molecyle group surface area
- real, dimension(:), allocatable **qi**

  [-] (nc) Combinatorial term param
- real, dimension(:), allocatable **ri**

  [-] (nc) Combinatorial term param

6.137.1 Detailed Description

Structure for active unifac parameters.
The documentation for this type was generated from the following file:

- unifac.f90

6.138 unifacdata::unifacprm Type Reference

Public Attributes

- integer **subgrp**
- integer **maingrp**
- character(len=uid_len) **formula**
- real **rk**
- real **qk**
- character(len=ref_len) **datasource**

The documentation for this type was generated from the following file:

- unifacdata.f90

6.139 unifacdata::unifacuij Type Reference

Public Attributes

- integer **mgi**
- integer **mgj**
- real **aij**

  [K]
- real **bij**

  [-]
- real **cij**

  [1/K]
- character(len=ref_len) **datasource**

The documentation for this type was generated from the following file:

- unifacdata.f90

6.140 stringmod::value Interface Reference

Public Member Functions

- subroutine **value_dr** (str, rnum, ios)
- subroutine **value_sr** (str, rnum, ios)
- subroutine **value_di** (str, inum, ios)
- subroutine **value_si** (str, inum, ios)

The documentation for this interface was generated from the following file:

- stringmod.f90

6.141 stringmod::writenum Interface Reference

Public Member Functions

- subroutine **write_dr** (rnum, str, fmt)
- subroutine **write_sr** (rnum, str, fmt)
- subroutine **write_di** (inum, str, fmt)
- subroutine **write_si** (inum, str, fmt)

The documentation for this interface was generated from the following file:

- stringmod.f90

6.142 stringmod::writeq Interface Reference

Public Member Functions

- subroutine **writeq_dr** (unit, namestr, value, fmt)
- subroutine **writeq_sr** (unit, namestr, value, fmt)
- subroutine **writeq_di** (unit, namestr, ivalue, fmt)
- subroutine **writeq_si** (unit, namestr, ivalue, fmt)

The documentation for this interface was generated from the following file:

- stringmod.f90

Index