iapws Documentation

Release 1.5.2

Juan José Gómez Romera

Apr 12, 2022

Contents

CHAPTER 1

Introduction

Python implementation of standard from IAPWS (http://www.iapws.org/release.html).

- Home: https://github.com/jjgomera/iapws
- Author: Juan José Gómez Romera <jjgomera@gmail.com>
- License: GPL-3
- Documentation: http://iapws.readthedocs.io/

CHAPTER 2

Dependences

- python 2x, 3x, compatible with both versions
- Numpy-scipy: library with mathematic and scientific tools

CHAPTER 3

Installation

In debian you can find in official repositories in jessie, testing and sid. In ubuntu it's in official repositories from ubuntu saucy (13.10). In other system you can install using pip:

```
pip install iapws
```

or directly cloning the github repository:

```
git clone https://github.com/jjgomera/iapws.git
```

and adding the folder to a python path.

CHAPTER 4

Features

This module implements almost the full set of standards:

Releases:

- R1-76(2014): Revised Release on the Surface Tension of Ordynary Water Substance, *iapws._iapws._Tension()*
- R2-83(1992): Release on the Values of Temperature, Pressure and Density of Ordynary and Heavy Water Substances at their Respectives Critical Points, *iapws._iapws()*
- R5-85(1994): Release on Surface Tension of Heavy Water Substance, *iapws._iapws._D2O_Tension()*
- R6-95(2018): Revised Release on the IAPWS Formulation 1995 for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use, *iapws.iapws95.IAPWS95()*
- R7-97(2012): Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, *iapws.iapws97()*
- R8-97: Release on the Static Dielectric Constant of Ordinary Water Substance for Temperatures from 238 K to 873 K and Pressures up to 1000 MPa, *iapws._iapws._Dielectric()*
- R9-97: Release on the Refractive Index of Ordinary Water Substance as a Function of Wavelength, Temperature and Pressure, *iapws._iapws._Refractive()*
- R10-06(2009): Revised Release on the Equation of State 2006 for H2O Ice Ih, *iapws._iapws._Ice()*
- R11-07(2019): Release on the Ionization Constant of H2O, *iapws._iapws._Kw()*
- R12-08: Release on the IAPWS Formulation 2008 for the Viscosity of Ordinary Water Substance, *iapws._iapws._Viscosity()*
- R13-08: Release on the IAPWS Formulation 2008 for the Thermodynamic Properties of Seawater, *iapws.iapws08()*
- R14-08(2011): Revised Release on the Pressure along the Melting and Sublimation Curves of Ordinary Water Substance, *iapws._iapws._Melting_Pressure()*, *iapws._iapws._Sublimation_Pressure()*

- R15-11: Release on the IAPWS Formulation 2011 for the Thermal Conductivity of Ordinary Water Substance, *iapws._iapws._ThCond()*
- R16-17(2018): Release on the IAPWS Formulation 2017 for the Thermodynamic Properties of Heavy Water, *iapws.iapws95.D2O()*
- R17-20: Release on the IAPWS Formulation 2020 for the Viscosity of Heavy Water, *iapws._iapws._D2O_Viscosity()*
- R18-21: Release on the IAPWS Formulation 2021 for the Thermal Conductivity of Heavy Water, *iapws._iapws._D2O_ThCond()*

Supplementary Releases:

- SR1-86(1992): Revised Supplementary Release on Saturation Properties of Ordinary Water Substance, *iapws.iapws95.MEoS._Liquid_Density()*, *iapws.iapws95.MEoS._Vapor_Density()*, *iapws.iapws95.MEoS._Vapor_Pressure()*
- SR2-01(2014): Revised Supplementary Release on Backward Equations for Pressure as a Function of Enthalpy and Entropy p(h,s) for Regions 1 and 2 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, *iapws.iapws97._Backward1_P_hs()*, *iapws.iapws97._Backward2_P_hs()*
- SR3-03(2014): Revised Supplementary Release on Backward Equations for the Functions T(p,h), v(p,h), and T(p,s), v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, *iapws.iapws97._Backward3_T_Ph()*, *iapws.iapws97._Backward3_T_Ps()*, *iapws.iapws97._Backward3_v_Ph()*, *iapws.iapws97._Backward3_v_Ps()*
- SR4-04(2014): Revised Supplementary Release on Backward Equations p(h,s) for Region 3, Equations as a Function of h and s for the Region Boundaries, and an Equation Tsat(h,s) for Region 4 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, *iapws.iapws97._Backward3_P_hs()*
- SR5-05(2016): Revised Supplementary Release on Backward Equations for Specific Volume as a Function of Pressure and Temperature v(p,T) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, *iapws.iapws97._Backward3_v_PT()*
- SR6-08(2011): Revised Supplementary Release on Properties of Liquid Water at 0.1 MPa, *iapws._iapws._Liquid()*
- SR7-09: Supplementary Release on a Computationally Efficient Thermodynamic Formulation for Liquid Water for Oceanographic Use, *iapws.iapws08.SeaWater._waterSupp()*

Guidelines:

- G1-90: Electrolytic Conductivity (Specific Conductance) of Liquid and Dense Supercritical Water from 0°C to 800°C and Pressures up to 1000 MPa, *iapws._iapws._Conductivity()*
- G2-90(1994): Solubility of Sodium Sulfate in Aqueous Mixtures of Sodium Chloride and Sulfuric Acid from Water to Concentrated Solutions, from 250 °C to 350 °C, *iapws.iapws08._solNa2SO4()*
- G3-00(2012): Revised Guideline on the Critical Locus of Aqueous Solutions of Sodium Chloride, *iapws.iapws08._critNaCl()*
- G4-01: Guideline on the IAPWS Formulation 2001 for the Thermodynamic Properties of Ammonia-Water Mixtures, *iapws.ammonia()*
- G5-01(2016): Guideline on the Use of Fundamental Physical Constants and Basic Constants of Water, *iapws._iapws()*
- G6-03: Guideline on the Tabular Taylor Series Expansion (TTSE) Method for Calculation of Thermodynamic Properties of Water and Steam Applied to IAPWS-95 as an Example (Not implemented)

- G7-04: Guideline on the Henry's Constant and Vapor-Liquid Distribution Constant for Gases in H2O and D2O at High Temperatures, *iapws._iapws._Henry()*, *iapws._iapws._Kvalue()*
- G8-10: Guideline on an Equation of State for Humid Air in Contact with Seawater and Ice, Consistent with the IAPWS Formulation 2008 for the Thermodynamic Properties of Seawater, *iapws.humidAir.HumidAir()*
- G9-12: Guideline on a Low-Temperature Extension of the IAPWS-95 Formulation for Water Vapor, *iapws.iapws95.IAPWS95._phiex()*
- G10-15: Guideline on the Thermal Conductivity of Seawater, *iapws.iapws08._ThCond_SeaWater()*
- G11-15: Guideline on a Virial Equation for the Fugacity of H2O in Humid Air, *iapws.humidAir._virial()*
- G12-15: Guideline on Thermodynamic Properties of Supercooled Water, *iapws._iapws._Supercooled()*
- G13-15: Guideline on the Fast Calculation of Steam and Water Properties with the Spline-Based Table Look-Up Method (SBTL) (Not implemented)
- G14-19: Guideline on the Surface Tension of Seawater, *iapws.iapws08._Tension_SeaWater()*

Advisory Notes:

- AN1-03: Uncertainties in Enthalpy for the IAPWS Formulation 1995 for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use (IAPWS-95) and the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam (IAPWS-IF97)
- AN2-04(2013): Role of Various IAPWS Documents Concerning the Thermodynamic Properties of Ordinary Water Substance
- AN3-07(2018): Thermodynamic Derivatives from IAPWS Formulations, *iapws._utils.deriv_G()*, *iapws._utils.deriv_H()*
- AN4-09: Roles of IAPWS and CIPM Standards for the Density of Water
- AN5-13(2016): Industrial Calculation of the Thermodynamic Properties of Seawater, `iapws.iapws08.Seawater._waterIF97()`, *iapws.iapws08._Tb()*, *iapws.iapws08._Tf()*, *iapws.iapws08._Triple()*, *iapws.iapws08._OsmoticPressure()*
- AN6-16: Relationship between Various IAPWS Documents and the International Thermodynamic Equation of Seawater - 2010 (TEOS-10)

CHAPTER 5

Documentation

You can navigate the full documentation of package:

5.1 iapws package

5.1.1 iapws._iapws module

Miscelaneous IAPWS standards. This module include:

- *_Ice()*: Ice Ih state equation
- *_Liquid()*: Properties of liquid water at 0.1 MPa
- *_Supercooled()*: Thermodynamic properties of supercooled water
- *_Sublimation_Pressure()*: Sublimation pressure correlation
- *_Melting_Pressure()*: Melting pressure correlation
- *_Viscosity()*: Viscosity correlation
- *_ThCond()*: Themal conductivity correlation
- *_Tension()*: Surface tension correlation
- *_Dielectric()*: Dielectric constant correlation
- *_Refractive()*: Refractive index correlation
- *_Kw()*: Ionization constant correlation for ordinary water
- *_Conductivity()*: Electrolytic conductivity correlation
- *_D2O_Viscosity()*: Viscosity correlation for heavy water
- *_D2O_ThCond()*: Thermal conductivity correlation for heavy water
- *_D2O_Tension()*: Surface tension correlation for heavy water

- *_D2O_Sublimation_Pressure()*: Sublimation Pressure correlation for heavy water
- *_D2O_Melting_Pressure()*: Melting Pressure correlation for heavy water
- *_Henry()*: Henry constant for liquid-gas equilibrium
- *_Kvalue()*: Vapor-liquid distribution constant

iapws._iapws._Ice(*T*, *P*)

Basic state equation for Ice Ih

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]

Returns

prop –

Dict with calculated properties of ice. The available properties are:

- rho: Density, [kg/m^3]
- h: Specific enthalpy, [kJ/kg]
- u: Specific internal energy, [kJ/kg]
- a: Specific Helmholtz energy, [kJ/kg]
- g: Specific Gibbs energy, [kJ/kg]
- s: Specific entropy, [kJ/kgK]
- cp: Specific isobaric heat capacity, [kJ/kgK]
- alfav: Cubic expansion coefficient, [1/K]
- beta: Pressure coefficient, [MPa/K]
- xkappa: Isothermal compressibility, [1/MPa]
- ks: Isentropic compressibility, [1/MPa]
- gt: [g/T]P
- gtt: [^{2}g/T^2]P
- gp: [g/P]T
- gpp: [^{2}g/P^2]T
- gtp: [^{2}g/TP]

Return type dict

Notes

Raise NotImplementedError if input isn't in limit:

- T 273.16
- P 208.566
- State below the melting and sublimation lines

Examples

```
>>> st1 = _Ice(100, 100)
>>> st1["rho"], st1["h"], st1["s"]
941.678203297 -483.491635676 -2.61195122589
```

```
>>> st2 = _Ice(273.152519,0.101325)
>>> st2["a"], st2["u"], st2["cp"]
-0.00918701567 -333.465403393 2.09671391024
```

```
>>> st3 = _Ice(273.16,611.657e-6)
>>> st3["alfav"], st3["beta"], st3["xkappa"], st3["ks"]
0.000159863102566 1.35714764659 1.17793449348e-04 1.14161597779e-04
```

References

IAPWS, Revised Release on the Equation of State 2006 for H2O Ice Ih September 2009, http://iapws.org/relguide/Ice-2009.html

iapws._iapws.**_Liquid**(*T*, *P=0.1*)

Supplementary release on properties of liquid water at 0.1 MPa

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa] Although this relation is for P=0.1MPa, can be extrapoled at pressure 0.3 MPa

Returns

prop –

Dict with calculated properties of water. The available properties are:

- h: Specific enthalpy, [kJ/kg]
- u: Specific internal energy, [kJ/kg]
- a: Specific Helmholtz energy, [kJ/kg]
- g: Specific Gibbs energy, [kJ/kg]
- s: Specific entropy, [kJ/kgK]
- cp: Specific isobaric heat capacity, [kJ/kgK]
- cv: Specific isochoric heat capacity, [kJ/kgK]
- w: Speed of sound, [m/s^2]
- rho: Density, [kg/m^3]
- v: Specific volume, [m^3/kg]
- vt: [v/T]P, [m^3/kgK]
- vtt: [^{2}v/T^2]P, [m^3/kgK^2]
- vp: [v/P]T, [m^3/kg/MPa]
- vtp: [^{2}v/TP], [m^3/kg/MPa]
- alfav: Cubic expansion coefficient, [1/K]

- xkappa : Isothermal compressibility, [1/MPa]
- ks: Isentropic compressibility, [1/MPa]
- mu: Viscosity, [mPas]
- k: Thermal conductivity, [W/mK]
- epsilon: Dielectric constant, [-]

Return type dict

Notes

Raise `NotImplementedError` if input isn't in limit:

- 253.15 T 383.15
- 0.1 P 0.3

Examples

```
>>> st1 = _Liquid(260)
>>> st1["rho"], st1["h"], st1["s"]
997.0683602710492 -55.86223174460868 -0.20998554842619535
```

References

IAPWS, Revised Supplementary Release on Properties of Liquid Water at 0.1 MPa, http://www.iapws.org/relguide/LiquidWater.html

iapws._iapws._**Supercooled**(*T*, *P*)

Guideline on thermodynamic properties of supercooled water

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]

Returns

prop –

Dict with calculated properties of water. The available properties are:

- L: Ordering field, [-]
- x: Mole fraction of low-density structure, [-]
- rho: Density, [kg/m^3]
- s: Specific entropy, [kJ/kgK]
- h: Specific enthalpy, [kJ/kg]
- u: Specific internal energy, [kJ/kg]
- a: Specific Helmholtz energy, [kJ/kg]
- g: Specific Gibbs energy, [kJ/kg]
- alfap: Thermal expansion coefficient, [1/K]

- xkappa : Isothermal compressibility, [1/MPa]
- cp: Specific isobaric heat capacity, [kJ/kgK]
- cv: Specific isochoric heat capacity, [kJ/kgK]
- w: Speed of sound, [m/s^2]

Return type dict

Notes

Raise `NotImplementedError` if input isn't in limit:

- Tm T 300
- 0 < P 1000

The minimum temperature in range of validity is the melting temperature, it depend of pressure

Raise `RuntimeError` if solution isn't founded

Examples

```
>>> liq = _supercooled(235.15, 0.101325)
>>> liq["rho"], liq["cp"], liq["w"]
968.09999 5.997563 1134.5855
```

References

iapws, guideline on thermodynamic properties of supercooled water, http://iapws.org/relguide/Supercooled.html

`iapws._iapws.`**`_Sublimation_Pressure`**`(T)`

Sublimation Pressure correlation

Parameters **T** (*float*) – Temperature, [K]

Returns **P** – Pressure at sublimation line, [MPa]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- 50 T 273.16

Examples

```
>>> _Sublimation_Pressure(230)
8.947352740189152e-06
```

References

IAPWS, Revised Release on the Pressure along the Melting and Sublimation Curves of Ordinary Water Substance, http://iapws.org/relguide/MeltSub.html.

iapws._iapws.**_Melting_Pressure**(*T*, *ice='Ih'*)
Melting Pressure correlation

Parameters

- **T** (*float*) – Temperature, [K]
- **ice** (*string*) – Type of ice: Ih, III, V, VI, VII. Below 273.15 is a mandatory input, the ice Ih is the default value. Above 273.15, the ice type is unnecesary.

Returns **P** – Pressure at sublimation line, [MPa]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- 251.165 T 715

Examples

```
>>> _Melting_Pressure(260)
8.947352740189152e-06
>>> _Melting_Pressure(254, "III")
268.6846466336108
```

References

IAPWS, Revised Release on the Pressure along the Melting and Sublimation Curves of Ordinary Water Substance, http://iapws.org/relguide/MeltSub.html.

iapws._iapws.**_Viscosity**(*rho*, *T*, *fase=None*, *drho=None*)
Equation for the Viscosity

Parameters

- **rho** (*float*) – Density, [kg/m^3]
- **T** (*float*) – Temperature, [K]
- **fase** (*dict, optional for calculate critical enhancement*) – phase properties
- **drho** (*float, optional for calculate critical enhancement*) – [ρ/P]T at reference state,

Returns μ – Viscosity, [Pa·s]

Return type float

Examples

```
>>> _Viscosity(998, 298.15)
0.0008897351001498108
>>> _Viscosity(600, 873.15)
7.743019522728247e-05
```

References

IAPWS, Release on the IAPWS Formulation 2008 for the Viscosity of Ordinary Water Substance, http://www.iapws.org/relguide/viscosity.html

iapws._iapws._**ThCond**(*rho*, *T*, *fase=None*, *drho=None*)

Equation for the thermal conductivity

Parameters

- **rho** (*float*) – Density, [kg/m^3]
- **T** (*float*) – Temperature, [K]
- **fase** (*dict, optional for calculate critical enhancement*) – phase properties
- **drho** (*float, optional for calculate critical enhancement*) – [ρ/P]T at reference state,

Returns **k** – Thermal conductivity, [W/mK]

Return type float

Examples

```
>>> _ThCond(998, 298.15)
0.6077128675880629
>>> _ThCond(0, 873.15)
0.07910346589648833
```

References

IAPWS, Release on the IAPWS Formulation 2011 for the Thermal Conductivity of Ordinary Water Substance, http://www.iapws.org/relguide/ThCond.html

iapws._iapws._**Tension**(*T*)

Equation for the surface tension

Parameters **T** (*float*) – Temperature, [K]

Returns σ – Surface tension, [N/m]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- 248.15 T 647

- Estrapolate to -25ºC in supercooled liquid metastable state

Examples

```
>>> _Tension(300)
0.0716859625
>>> _Tension(450)
0.0428914992
```

References

IAPWS, Revised Release on Surface Tension of Ordinary Water Substance June 2014, http://www.iapws.org/relguide/Surf-H2O.html

iapws._iapws._**Dielectric**(*rho*, *T*)

Equation for the Dielectric constant

Parameters

- **rho** (*float*) – Density, [kg/m^3]
- **T** (*float*) – Temperature, [K]

Returns **epsilon** – Dielectric constant, [-]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- 238 T 1200

Examples

```
>>> _Dielectric(999.242866, 298.15)
78.5907250
>>> _Dielectric(26.0569558, 873.15)
1.12620970
```

References

IAPWS, Release on the Static Dielectric Constant of Ordinary Water Substance for Temperatures from 238 K to 873 K and Pressures up to 1000 MPa, http://www.iapws.org/relguide/Dielec.html

iapws._iapws._**Refractive**(*rho*, *T*, *lr=0.5893*)

Equation for the refractive index

Parameters

- **rho** (*float*) – Density, [kg/m^3]
- **T** (*float*) – Temperature, [K]
- **lr** (*float, optional*) – Light Wavelength, [μm]

Returns **n** – Refractive index, [-]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- 0 ρ 1060
- 261.15 T 773.15
- 0.2 λ 1.1

Examples

```
>>> _Refractive(997.047435, 298.15, 0.2265)
1.39277824
>>> _Refractive(30.4758534, 773.15, 0.5893)
1.00949307
```

References

IAPWS, Release on the Refractive Index of Ordinary Water Substance as a Function of Wavelength, Temperature and Pressure, http://www.iapws.org/relguide/rindex.pdf

`iapws._iapws._Kw`(*rho*, *T*)

Equation for the ionization constant of ordinary water

Parameters

- **rho** (*float*) – Density, [kg/m^3]
- **T** (*float*) – Temperature, [K]

Returns **pKw** – Ionization constant in -log10(kw), [-]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- 0 ρ 1250
- 273.15 T 1073.15

Examples

```
>>> _Kw(1000, 300)
13.906565
```

References

IAPWS, Release on the Ionization Constant of H2O, http://www.iapws.org/relguide/Ionization.pdf

iapws._iapws._**Conductivity**(*rho*, *T*)

Equation for the electrolytic conductivity of liquid and dense supercrítical water

Parameters

- **rho** (*float*) – Density, [kg/m^3]
- **T** (*float*) – Temperature, [K]

Returns **K** – Electrolytic conductivity, [S/m]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- 600 ρ 1200
- 273.15 T 1073.15

Examples

```
>>> _Conductivity(1000, 373.15)
1.13
```

References

IAPWS, Electrolytic Conductivity (Specific Conductance) of Liquid and Dense Supercritical Water from 0°C to 800°C and Pressures up to 1000 MPa, http://www.iapws.org/relguide/conduct.pdf

iapws._iapws._**D2O_Viscosity**(*rho*, *T*, *fase=None*, *drho=None*)

Equation for the Viscosity of heavy water

Parameters

- **rho** (*float*) – Density, [kg/m^3]
- **T** (*float*) – Temperature, [K]
- **fase** (*dict, optional for calculate critical enhancement*) – phase properties
- **drho** (*float, optional for calculate critical enhancement*) – [ρ/P]T at reference state,

Returns μ – Viscosity, [Pa·s]

Return type float

Examples

```
>>> _D2O_Viscosity(998, 298.15)
0.0008897351001498108
>>> _D2O_Viscosity(600, 873.15)
7.743019522728247e-05
```

References

IAPWS, Release on the IAPWS Formulation 2020 for the Viscosity of Heavy Water, http://iapws.org/relguide/D2Ovisc.pdf

iapws._iapws.**_D2O_ThCond**(*rho*, *T*, *fase=None*, *drho=None*)

Equation for the thermal conductivity of heavy water

Parameters

- **rho** (*float*) – Density, [kg/m^3]
- **T** (*float*) – Temperature, [K]
- **fase** (*dict, optional for calculate critical enhancement*) – phase properties
- **drho** (*float, optional for calculate critical enhancement*) – [ρ/P]T at reference state,

Returns **k** – Thermal conductivity, [W/mK]

Return type float

Examples

```
>>> _D2O_ThCond(998, 298.15)
0.6077128675880629
>>> _D2O_ThCond(0, 873.15)
0.07910346589648833
```

References

IAPWS, Release on the IAPWS Formulation 2021 for the Thermal Conductivity of Heavy Water, http://iapws.org/relguide/D2OThCond.pdf

iapws._iapws.**_D2O_Tension**(*T*)

Equation for the surface tension of heavy water

Parameters **T** (*float*) – Temperature, [K]

Returns σ – Surface tension, [N/m]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- 269.65 T 643.847

Examples

```
>>> _D2O_Tension(298.15)
0.07186
>>> _D2O_Tension(573.15)
0.01399
```

References

IAPWS, Release on Surface Tension of Heavy Water Substance, http://www.iapws.org/relguide/surfd2o.pdf

iapws._iapws.**_D2O_Sublimation_Pressure**(*T*)

Sublimation Pressure correlation for heavy water

Parameters **T** (*float*) – Temperature, [K]

Returns **P** – Pressure at sublimation line, [MPa]

Return type float

Notes

Raise NotImplementedError if input isn't in limit:

- 210 T 276.969

Examples

```
>>> _Sublimation_Pressure(245)
3.27390934e-5
```

References

IAPWS, Revised Release on the IAPWS Formulation 2017 for the Thermodynamic Properties of Heavy Water, http://www.iapws.org/relguide/Heavy.html.

iapws._iapws.**_D2O_Melting_Pressure**(*T*, *ice='Ih'*)

Melting Pressure correlation for heavy water

Parameters

- **T** (*float*) – Temperature, [K]
- **ice** (*string*) – Type of ice: Ih, III, V, VI, VII. Below 276.969 is a mandatory input, the ice Ih is the default value. Above 276.969, the ice type is unnecesary.

Returns **P** – Pressure at melting line, [MPa]

Return type float

Notes

Raise NotImplementedError if input isn't in limit:

- 254.415 T 315

Examples

```
>>> _D2O__Melting_Pressure(260)
8.947352740189152e-06
>>> _D2O__Melting_Pressure(254, "III")
268.6846466336108
```

References

IAPWS, Revised Release on the Pressure along the Melting and Sublimation Curves of Ordinary Water Substance, http://iapws.org/relguide/MeltSub.html.

iapws._iapws.**_Henry**(*T*, *gas*, *liquid='H2O'*)
Equation for the calculation of Henry's constant

Parameters

- **T** (*float*) – Temperature, [K]
- **gas** (*string*) – Name of gas to calculate solubility
- **liquid** (*string*) – Name of liquid solvent, can be H20 (default) or D2O

Returns **kw** – Henry's constant, [MPa]

Return type float

Notes

The gas availables for H2O solvent are He, Ne, Ar, Kr, Xe, H2, N2, O2, CO, CO2, H2S, CH4, C2H6, SF6 For D2O as solvent He, Ne, Ar, Kr, Xe, D2, CH4

Raise `NotImplementedError` if input gas or liquid are unsupported

Examples

```
>>> _Henry(500, "He")
1.1973
>>> _Henry(300, "D2", "D2O")
1.6594
```

References

IAPWS, Guideline on the Henry's Constant and Vapor-Liquid Distribution Constant for Gases in H2O and D2O at High Temperatures, http://www.iapws.org/relguide/HenGuide.html

iapws._iapws.**_Kvalue**(*T*, *gas*, *liquid='H2O'*)
Equation for the vapor-liquid distribution constant

Parameters

- **T** (*float*) – Temperature, [K]
- **gas** (*string*) – Name of gas to calculate solubility
- **liquid** (*string*) – Name of liquid solvent, can be H20 (default) or D2O

Returns **kd** – Vapor-liquid distribution constant, [-]

Return type float

Notes

The gas availables for H2O solvent are He, Ne, Ar, Kr, Xe, H2, N2, O2, CO, CO2, H2S, CH4, C2H6, SF6

For D2O as solvent He, Ne, Ar, Kr, Xe, D2, CH4

Raise `NotImplementedError` if input gas or liquid are unsupported

Examples

```
>>> _Kvalue(600, "He")
3.8019
>>> _Kvalue(300, "D2", "D2O")
14.3520
```

References

IAPWS, Guideline on the Henry's Constant and Vapor-Liquid Distribution Constant for Gases in H2O and D2O at High Temperatures, http://www.iapws.org/relguide/HenGuide.html

5.1.2 iapws._utils module

Miscelaneous internal utilities. This module include:

- *getphase()*: Get phase string of state
- *_fase*: Base class to define a phase state
- *deriv_H()*: Calculate generic partial derivative with a fundamental Helmholtz free energy equation of state
- *deriv_G()*: Calculate generic partial derivative with a fundamental Gibbs free energy equation of state

iapws._utils.**getphase**(*Tc*, *Pc*, *T*, *P*, *x*, *region*)
Return fluid phase string name

Parameters

- **Tc** (*float*) – Critical temperature, [K]
- **Pc** (*float*) – Critical pressure, [MPa]
- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]
- **x** (*float*) – Quality, [-]
- **region** (*int*) – Region number, used only for IAPWS97 region definition

Returns **phase** – Phase name

Return type str

class iapws._utils.**_fase**
Class to implement a null phase

Attributes

Gruneisen

IntP

Ks

Kt

Prandt

Z

Z_rho

a

alfa

alfap

alfav

betap

betas

cp

cp_cv

cv

dhdP_T

dhdP_rho

dhdT_P

dhdT_rho

dhdrho_P

dhdrho_T

dpdT_rho

dpdrho_T

drhodP_T

drhodT_P

epsilon

f

fi

g

gamma

h

hInput

joule

k

kappa

ks

kt

mu

n

nu

rho

s

u

v

w

iapws._utils.**deriv_H**(*state*, *z*, *x*, *y*, *fase*)

Calculate generic partial derivative $\frac{\partial z}{\partial x}\Big|_y$ from a fundamental helmholtz free energy equation of state

Parameters

- **state** (*any python object*) – Only need to define P and T properties, non phase specific properties
- **z** (*str*) – Name of variables in numerator term of derivatives
- **x** (*str*) – Name of variables in denominator term of derivatives
- **y** (*str*) – Name of constant variable in partial derivaritive
- **fase** (*any python object*) – Define phase specific properties (v, cv, alfap, s, betap)

Notes

x, y and z can be the following values:

- P: Pressure
- T: Temperature
- v: Specific volume
- rho: Density
- u: Internal Energy
- h: Enthalpy
- s: Entropy
- g: Gibbs free energy
- a: Helmholtz free energy

Returns **deriv** – z/x|y

Return type float

References

IAPWS, Revised Advisory Note No. 3: Thermodynamic Derivatives from IAPWS Formulations, http://www.iapws.org/relguide/Advise3.pdf

iapws._utils.**deriv_G**(*state*, *z*, *x*, *y*, *fase*)
Calculate generic partial derivative $\frac{\partial z}{\partial x}\Big|_y$ from a fundamental Gibbs free energy equation of state

Parameters

- **state** (*any python object*) – Only need to define P and T properties, non phase specific properties
- **z** (*str*) – Name of variables in numerator term of derivatives
- **x** (*str*) – Name of variables in denominator term of derivatives
- **y** (*str*) – Name of constant variable in partial derivaritive
- **fase** (*any python object*) – Define phase specific properties (v, cp, alfav, s, xkappa)

Notes

x, y and z can be the following values:

- P: Pressure
- T: Temperature
- v: Specific volume
- rho: Density
- u: Internal Energy
- h: Enthalpy
- s: Entropy
- g: Gibbs free energy
- a: Helmholtz free energy

Returns **deriv** – z/x|y

Return type float

References

IAPWS, Revised Advisory Note No. 3: Thermodynamic Derivatives from IAPWS Formulations, http://www.iapws.org/relguide/Advise3.pdf

5.1.3 iapws.iapws95 module

Implemented multiparameter equation of state as a Helmholtz free energy:

- *MEoS*: Base class of multiparameter equation of state
- *IAPWS95*: 2016 revision of 1995 formulation for ordinaty water
- *D2O*: 2017 formulation for heavy water.

iapws.iapws95.**_phir**(*tau*, *delta*, *coef*)
Residual contribution to the adimensional free Helmholtz energy

Parameters

- **tau** (*float*) – Inverse reduced temperature Tc/T, [-]
- **delta** (*float*) – Reduced density rho/rhoc, [-]
- **coef** (*dict*) – Dictionary with equation of state parameters

Returns **fir** – Adimensional free Helmholtz energy

Return type float

References

IAPWS, Revised Release on the IAPWS Formulation 1995 for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use, September 2016, Table 5 http://www.iapws.org/relguide/IAPWS-95.html

iapws.iapws95.**_phird**(*tau*, *delta*, *coef*)
Residual contribution to the adimensional free Helmholtz energy, delta derivative

Parameters

- **tau** (*float*) – Inverse reduced temperature Tc/T, [-]
- **delta** (*float*) – Reduced density rho/rhoc, [-]
- **coef** (*dict*) – Dictionary with equation of state parameters

Returns

fird –

$$\left.\frac{\partial \phi_\delta^r}{\partial \delta}\right|_\tau$$

Return type float

References

IAPWS, Revised Release on the IAPWS Formulation 1995 for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use, September 2016, Table 5 http://www.iapws.org/relguide/IAPWS-95.html

iapws.iapws95.**_phirt**(*tau*, *delta*, *coef*)
Residual contribution to the adimensional free Helmholtz energy, tau derivative

Parameters

- **tau** (*float*) – Inverse reduced temperature Tc/T, [-]
- **delta** (*float*) – Reduced density rho/rhoc, [-]
- **coef** (*dict*) – Dictionary with equation of state parameters

Returns

firt –

$$\left.\frac{\partial \phi_\tau^r}{\partial \tau}\right|_\delta$$

Return type float

References

IAPWS, Revised Release on the IAPWS Formulation 1995 for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use, September 2016, Table 5 http://www.iapws.org/relguide/IAPWS-95.html

class iapws.iapws95.**MEoS**(***kwargs*)

General implementation of multiparameter equation of state. From this derived all child class specified per individual compounds

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]
- **rho** (*float*) – Density, [kg/m^3]
- **v** (*float*) – Specific volume, [m^3/kg]
- **h** (*float*) – Specific enthalpy, [kJ/kg]
- **s** (*float*) – Specific entropy, [kJ/kgK]
- **u** (*float*) – Specific internal energy, [kJ/kg]
- **x** (*float*) – Vapor quality, [-]
- **l** (*float, optional*) – Wavelength of light, for refractive index, [μm]
- **rho0** (*float, optional*) – Initial value of density, to improve iteration, [kg/m^3]
- **T0** (*float, optional*) – Initial value of temperature, to improve iteration, [K]
- **x0** (*Initial value of vapor quality, necessary in bad input pair definition*) – where there are two valid solution (T-h, T-s)

Notes

- It needs two incoming properties of T, P, rho, h, s, u.
- v as a alternate input parameter to rho
- T-x, P-x, preferred input pair to specified a point in two phases region

The calculated instance has the following properties:

- P: Pressure, [MPa]
- T: Temperature, [K]
- x: Vapor quality, [-]
- g: Specific Gibbs free energy, [kJ/kg]
- a: Specific Helmholtz free energy, [kJ/kg]
- v: Specific volume, [m^3/kg]
- r: Density, [kg/m^3]
- h: Specific enthalpy, [kJ/kg]

- u: Specific internal energy, [kJ/kg]
- s: Specific entropy, [kJ/kg·K]
- cp: Specific isobaric heat capacity, [kJ/kg·K]
- cv: Specific isochoric heat capacity, [kJ/kg·K]
- cp_cv: Heat capacity ratio, [-]
- Z: Compression factor, [-]
- fi: Fugacity coefficient, [-]
- f: Fugacity, [MPa]
- gamma: Isoentropic exponent, [-]
- alfav: Isobaric cubic expansion coefficient, [1/K]
- kappa: Isothermal compressibility, [1/MPa]
- kappas: Adiabatic compresibility, [1/MPa]
- alfap: Relative pressure coefficient, [1/K]
- betap: Isothermal stress coefficient, [kg/m^3]
- joule: Joule-Thomson coefficient, [K/MPa]
- betas: Isoentropic temperature-pressure coefficient, [-]
- Gruneisen: Gruneisen parameter, [-]
- virialB: Second virial coefficient, [m^3/kg]
- virialC: Third virial coefficient, [m^6/kg^2]
- dpdT_rho: Derivatives, dp/dT at constant rho, [MPa/K]
- dpdrho_T: Derivatives, dp/drho at constant T, [MPa·m^3/kg]
- drhodT_P: Derivatives, drho/dT at constant P, [kg/m^3·K]
- drhodP_T: Derivatives, drho/dP at constant T, [kg/m^3·MPa]
- dhdT_rho: Derivatives, dh/dT at constant rho, [kJ/kg·K]
- dhdP_T: Isothermal throttling coefficient, [kJ/kg·MPa]
- dhdT_P: Derivatives, dh/dT at constant P, [kJ/kg·K]
- dhdrho_T: Derivatives, dh/drho at constant T, [kJ·m^3/kg^2]
- dhdrho_P: Derivatives, dh/drho at constant P, [kJ·m^3/kg^2]
- dhdP_rho: Derivatives, dh/dP at constant rho, [kJ/kg·MPa]
- kt: Isothermal Expansion Coefficient, [-]
- ks: Adiabatic Compressibility, [1/MPa]
- Ks: Adiabatic bulk modulus, [MPa]
- Kt: Isothermal bulk modulus, [MPa]
- v0: Ideal specific volume, [m^3/kg]
- rho0: Ideal gas density, [kg/m^3]
- u0: Ideal specific internal energy, [kJ/kg]

- h0: Ideal specific enthalpy, [kJ/kg]
- s0: Ideal specific entropy, [kJ/kg·K]
- a0: Ideal specific Helmholtz free energy, [kJ/kg]
- g0: Ideal specific Gibbs free energy, [kJ/kg]
- cp0: Ideal specific isobaric heat capacity, [kJ/kg·K]
- cv0: Ideal specific isochoric heat capacity, [kJ/kg·K]
- w0: Ideal speed of sound, [m/s]
- gamma0: Ideal isoentropic exponent, [-]
- w: Speed of sound, [m/s]
- mu: Dynamic viscosity, [Pa·s]
- nu: Kinematic viscosity, [m^2/s]
- k: Thermal conductivity, [W/m·K]
- alfa: Thermal diffusivity, [m^2/s]
- sigma: Surface tension, [N/m]
- epsilon: Dielectric constant, [-]
- n: Refractive index, [-]
- Prandt: Prandtl number, [-]
- Pr: Reduced Pressure, [-]
- Tr: Reduced Temperature, [-]
- Hvap: Vaporization heat, [kJ/kg]
- Svap: Vaporization entropy, [kJ/kg·K]
- Z_rho: $(Z-1)/\rho$, [m^3/kg]
- IntP: Internal pressure, [MPa]
- invT: Negative reciprocal temperature, [1/K]
- hInput: Specific heat input, [kJ/kg]

Attributes

CP

Gruneisen

IntP

Ks

Kt

Prandt

Z

Z_rho

a

alfa

alfap

alfav

betap

betas

calculable Check if inputs are enough to define state

cp

cp_cv

cv

dhdP_T

dhdP_rho

dhdT_P

dhdT_rho

dhdrho_P

dhdrho_T

dpdT_rho

dpdrho_T

drhodP_T

drhodT_P

epsilon

f

fi

g

gamma

h

hInput

joule

k

kappa

ks

kt

mu

n

nu

rho

s

u

v

w

Methods

`__call__`(**kwargs)	Make instance callable to can add input parameter one to one
calculo()	Calculate procedure
derivative(z, x, y, fase)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, rho, u, h, s, g, a
fill(fase, estado)	Fill phase properties

CP = None

_Pv = None

_rhoL = None

_rhoG = None

status = 0

msg = 'Undefined'

kwargs = {'P': 0.0, 'T': 0.0, 'T0': None, 'h': None, 'l': 0.5893, 'rho': 0.0, 'rho

calculable
Check if inputs are enough to define state

calculo()
Calculate procedure

fill(*fase*, *estado*)
Fill phase properties

derivative(*z*, *x*, *y*, *fase*)
Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, rho, u, h, s, g, a

_saturation(*T*)
Saturation calculation for two phase search

_Helmholtz(*rho*, *T*)
Calculated properties from helmholtz free energy and derivatives

Parameters

- **rho** (*float*) – Density, [kg/m^3]
- **T** (*float*) – Temperature, [K]

Returns

prop –

Dictionary with calculated properties:

- fir: [-]
- fird: fir/δ|τ
- firdd: ^{2}fir/δ^2|τ

- delta: Reducen density rho/rhoc, [-]
- P: Pressure, [kPa]
- h: Enthalpy, [kJ/kg]
- s: Entropy, [kJ/kgK]
- cv: Isochoric specific heat, [kJ/kgK]
- alfav: Thermal expansion coefficient, [1/K]
- betap: Isothermal compressibility, [1/kPa]

Return type dict

References

IAPWS, Revised Release on the IAPWS Formulation 1995 for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use, September 2016, Table 3 http://www.iapws.org/relguide/IAPWS-95.html

_prop0 (*rho*, *T*)
Ideal gas properties

_phi0 (*tau*, *delta*)
Ideal gas Helmholtz free energy and derivatives

Parameters

- **tau** (*float*) – Inverse reduced temperature Tc/T, [-]
- **delta** (*float*) – Reduced density rho/rhoc, [-]

Returns **prop** – fio, [-] fiot: fio/τ|δ fiod: fio/δ|τ fiott: ^{2}fio/τ^2|δ fiodt: ^{2}fio/$\tau\delta$ fiodd: ^{2}fio/δ^2|τ

Return type dictionary with ideal adimensional helmholtz energy and deriv

References

IAPWS, Revised Release on the IAPWS Formulation 1995 for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use, September 2016, Table 4 http://www.iapws.org/relguide/IAPWS-95.html

_phir (*tau*, *delta*)
Residual contribution to the free Helmholtz energy

Parameters

- **tau** (*float*) – Inverse reduced temperature Tc/T, [-]
- **delta** (*float*) – Reduced density rho/rhoc, [-]

Returns

prop –

Dictionary with residual adimensional helmholtz energy and deriv:

- fir
- firt: fir/τ|δ,x
- fird: fir/δ|τ,x

- firtt: ${}^2\text{fir}/\tau^2|\delta$,x
- firdt: ${}^2\text{fir}/\tau\delta|$x
- firdd: ${}^2\text{fir}/\delta^2|\tau$,x

Return type dict

References

IAPWS, Revised Release on the IAPWS Formulation 1995 for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use, September 2016, Table 5 http://www.iapws.org/relguide/IAPWS-95.html

_virial(*T*)
Virial coefficient

Parameters **T** (*float*) – Temperature [K]

Returns

prop –

Dictionary with residual adimensional helmholtz energy:

- B: fir/$\delta|\delta$->0
- C: ${}^2\text{fir}/\delta^2|\delta$->0

Return type dict

_derivDimensional(*rho*, *T*)
Calcule the dimensional form or Helmholtz free energy derivatives

Parameters

- **rho** (*float*) – Density, [kg/m^3]
- **T** (*float*) – Temperature, [K]

Returns

prop –

Dictionary with residual helmholtz energy and derivatives:

- fir, [kJ/kg]
- firt: fir/T|ρ, [kJ/kgK]
- ird: fir/ρ|T, [kJ/m^{3}kg^2]
- firtt: ${}^2\text{fir}/\text{T}^2|\rho$, [kJ/kgK^2]
- firdt: ${}^2\text{fir}/\text{T}\rho$, [kJ/m^{3}kg^{2}K]
- firdd: ${}^2\text{fir}/\rho^2|$T, [kJ/m^{6}kg]

Return type dict

References

IAPWS, Guideline on an Equation of State for Humid Air in Contact with Seawater and Ice, Consistent with the IAPWS Formulation 2008 for the Thermodynamic Properties of Seawater, Table 7, http://www.iapws.org/relguide/SeaAir.html

_surface(*T*)
Generic equation for the surface tension

Parameters **T** (*float*) – Temperature, [K]

Returns σ – Surface tension, [N/m]

Return type float

Notes

Need a _surf dict in the derived class with the parameters keys: sigma: coefficient exp: exponent

classmethod _Vapor_Pressure(*T*)
Auxiliary equation for the vapour pressure

Parameters **T** (*float*) – Temperature, [K]

Returns **Pv** – Vapour pressure, [Pa]

Return type float

References

IAPWS, Revised Supplementary Release on Saturation Properties of Ordinary Water Substance September 1992, http://www.iapws.org/relguide/Supp-sat.html, Eq.1

classmethod _Liquid_Density(*T*)
Auxiliary equation for the density of saturated liquid

Parameters **T** (*float*) – Temperature, [K]

Returns **rho** – Saturated liquid density, [kg/m^3]

Return type float

References

IAPWS, Revised Supplementary Release on Saturation Properties of Ordinary Water Substance September 1992, http://www.iapws.org/relguide/Supp-sat.html, Eq.2

classmethod _Vapor_Density(*T*)
Auxiliary equation for the density of saturated vapor

Parameters **T** (*float*) – Temperature, [K]

Returns **rho** – Saturated vapor density, [kg/m^3]

Return type float

References

IAPWS, Revised Supplementary Release on Saturation Properties of Ordinary Water Substance September 1992, http://www.iapws.org/relguide/Supp-sat.html, Eq.3

classmethod _dPdT_sat(*T*)
Auxiliary equation for the dP/dT along saturation line

Parameters **T** (*float*) – Temperature, [K]

Returns **dPdT** – dPdT, [MPa/K]

Return type float

References

IAPWS, Revised Supplementary Release on Saturation Properties of Ordinary Water Substance September 1992, http://www.iapws.org/relguide/Supp-sat.html, derived from Eq.1

iapws.iapws95.**mainClassDoc**()
Function decorator used to automatic adiction of base class MEoS in subclass __doc__

class iapws.iapws95.**IAPWS95**(***kwargs*)
Implementation of IAPWS Formulation 1995 for ordinary water substance, (revised release of 2016), for internal procedures, see MEoS base class

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]
- **rho** (*float*) – Density, [kg/m^3]
- **v** (*float*) – Specific volume, [m^3/kg]
- **h** (*float*) – Specific enthalpy, [kJ/kg]
- **s** (*float*) – Specific entropy, [kJ/kgK]
- **u** (*float*) – Specific internal energy, [kJ/kg]
- **x** (*float*) – Vapor quality, [-]
- **l** (*float, optional*) – Wavelength of light, for refractive index, [μm]
- **rho0** (*float, optional*) – Initial value of density, to improve iteration, [kg/m^3]
- **T0** (*float, optional*) – Initial value of temperature, to improve iteration, [K]
- **x0** (*Initial value of vapor quality, necessary in bad input pair definition*) – where there are two valid solution (T-h, T-s)

Notes

- It needs two incoming properties of T, P, rho, h, s, u.
- v as a alternate input parameter to rho
- T-x, P-x, preferred input pair to specified a point in two phases region

The calculated instance has the following properties:

- P: Pressure, [MPa]
- T: Temperature, [K]
- x: Vapor quality, [-]
- g: Specific Gibbs free energy, [kJ/kg]
- a: Specific Helmholtz free energy, [kJ/kg]
- v: Specific volume, [m^3/kg]

- r: Density, [kg/m^3]
- h: Specific enthalpy, [kJ/kg]
- u: Specific internal energy, [kJ/kg]
- s: Specific entropy, [kJ/kg·K]
- cp: Specific isobaric heat capacity, [kJ/kg·K]
- cv: Specific isochoric heat capacity, [kJ/kg·K]
- cp_cv: Heat capacity ratio, [-]
- Z: Compression factor, [-]
- fi: Fugacity coefficient, [-]
- f: Fugacity, [MPa]
- gamma: Isoentropic exponent, [-]
- alfav: Isobaric cubic expansion coefficient, [1/K]
- kappa: Isothermal compressibility, [1/MPa]
- kappas: Adiabatic compresibility, [1/MPa]
- alfap: Relative pressure coefficient, [1/K]
- betap: Isothermal stress coefficient, [kg/m^3]
- joule: Joule-Thomson coefficient, [K/MPa]
- betas: Isoentropic temperature-pressure coefficient, [-]
- Gruneisen: Gruneisen parameter, [-]
- virialB: Second virial coefficient, [m^3/kg]
- virialC: Third virial coefficient, [m^6/kg^2]
- dpdT_rho: Derivatives, dp/dT at constant rho, [MPa/K]
- dpdrho_T: Derivatives, dp/drho at constant T, [MPa·m^3/kg]
- drhodT_P: Derivatives, drho/dT at constant P, [kg/m^3·K]
- drhodP_T: Derivatives, drho/dP at constant T, [kg/m^3·MPa]
- dhdT_rho: Derivatives, dh/dT at constant rho, [kJ/kg·K]
- dhdP_T: Isothermal throttling coefficient, [kJ/kg·MPa]
- dhdT_P: Derivatives, dh/dT at constant P, [kJ/kg·K]
- dhdrho_T: Derivatives, dh/drho at constant T, [kJ·m^3/kg^2]
- dhdrho_P: Derivatives, dh/drho at constant P, [kJ·m^3/kg^2]
- dhdP_rho: Derivatives, dh/dP at constant rho, [kJ/kg·MPa]
- kt: Isothermal Expansion Coefficient, [-]
- ks: Adiabatic Compressibility, [1/MPa]
- Ks: Adiabatic bulk modulus, [MPa]
- Kt: Isothermal bulk modulus, [MPa]
- v0: Ideal specific volume, [m^3/kg]

- rho0: Ideal gas density, [kg/m^3]
- u0: Ideal specific internal energy, [kJ/kg]
- h0: Ideal specific enthalpy, [kJ/kg]
- s0: Ideal specific entropy, [kJ/kg·K]
- a0: Ideal specific Helmholtz free energy, [kJ/kg]
- g0: Ideal specific Gibbs free energy, [kJ/kg]
- cp0: Ideal specific isobaric heat capacity, [kJ/kg·K]
- cv0: Ideal specific isochoric heat capacity, [kJ/kg·K]
- w0: Ideal speed of sound, [m/s]
- gamma0: Ideal isoentropic exponent, [-]
- w: Speed of sound, [m/s]
- mu: Dynamic viscosity, [Pa·s]
- nu: Kinematic viscosity, [m^2/s]
- k: Thermal conductivity, [W/m·K]
- alfa: Thermal diffusivity, [m^2/s]
- sigma: Surface tension, [N/m]
- epsilon: Dielectric constant, [-]
- n: Refractive index, [-]
- Prandt: Prandtl number, [-]
- Pr: Reduced Pressure, [-]
- Tr: Reduced Temperature, [-]
- Hvap: Vaporization heat, [kJ/kg]
- Svap: Vaporization entropy, [kJ/kg·K]
- Z_rho: $(Z-1)/\rho$, [m^3/kg]
- IntP: Internal pressure, [MPa]
- invT: Negative reciprocal temperature, [1/K]
- hInput: Specific heat input, [kJ/kg]

Examples

```
>>> water=IAPWS95(T=300, rho=996.5560)
>>> water.P, water.cv, water.w, water.s
0.0992418350 4.13018112 1501.51914 0.393062643
```

```
>>> water=IAPWS95(T=500, rho=0.435)
>>> water.P, water.cv, water.w, water.s
0.0999679423 1.50817541 548.31425 7.944882714
```

```
>>> water=IAPWS95(T=900., P=700)
>>> water.rho, water.cv, water.w, water.s
870.7690 2.66422350 2019.33608 4.17223802
```

```
>>> water=IAPWS95(T=300., P=0.1)
>>> water.P, water.rho, water.h, water.s, water.cp, water.w, water.virialB
0.10000 996.56 112.65 0.39306 4.1806 1501.5 -0.066682
```

```
>>> water=IAPWS95(T=500., P=0.1)
>>> water.P, water.rho, water.h, water.s, water.cp, water.w, water.virialB
0.10000 0.43514 2928.6 7.9447 1.9813 548.31 -0.0094137
```

```
>>> water=IAPWS95(T=450., x=0.5)
>>> water.T, water.P, water.rho, water.h, water.s, water.virialB
450.00 0.93220 9.5723 1761.8 4.3589 -0.013028
```

```
>>> water=IAPWS95(P=1.5, rho=1000.)
>>> water.T, water.rho, water.h, water.s, water.cp, water.w, water.virialB
286.44 1000.0 57.253 0.19931 4.1855 1462.1 -0.085566
```

```
>>> water=IAPWS95(h=3000, s=8.)
>>> water.T, water.P, water.h, water.s, water.cp, water.w, water.virialB
536.24 0.11970 3000.0 8.0000 1.9984 567.04 -0.0076606
```

```
>>> water=IAPWS95(h=150, s=0.4)
>>> water.T, water.P, water.rho, water.h, water.s, water.cp, water.w
301.27 35.50549 1011.48 150.00 0.40000 4.0932 1564.1
```

```
>>> water=IAPWS95(T=450., rho=300)
>>> water.T, water.P, water.rho, water.h, water.s, water.x, water.virialB
450.00 0.93220 300.00 770.82 2.1568 0.010693 -0.013028
```

```
>>> water=IAPWS95(rho=300., P=0.1)
>>> water.T, water.P, water.rho, water.h, water.s, water.x, water.virialB
372.76 0.10000 300.00 420.56 1.3110 0.0013528 -0.025144
```

```
>>> water=IAPWS95(h=1500., P=0.1)
>>> water.T, water.P, water.rho, water.h, water.s, water.x, water.virialB
372.76 0.10000 1.2303 1500.0 4.2068 0.47952 -0.025144
```

```
>>> water=IAPWS95(s=5., P=3.5)
>>> water.T, water.P, water.rho, water.h, water.s, water.x, water.virialB
515.71 3.5000 25.912 2222.8 5.0000 0.66921 -0.0085877
```

```
>>> water=IAPWS95(T=500., u=900)
>>> water.P, water.rho, water.u, water.h, water.s, water.cp, water.w
108.21 903.62 900.00 1019.8 2.4271 4.1751 1576.0
```

```
>>> water=IAPWS95(P=0.3, u=1550.)
>>> water.T, water.P, water.rho, water.u, water.h, water.s, water.x
406.67 0.30000 3.3029 1550.0 1640.8 4.3260 0.49893
```

```
>>> water=IAPWS95(rho=300, h=1000.)
>>> water.T, water.P, water.rho, water.u, water.h, water.s, water.x
494.92 2.3991 300.00 992.00 1000.0 2.6315 0.026071
```

```
>>> water=IAPWS95(rho=30, s=8.)
>>> water.T, water.P, water.rho, water.u, water.h, water.s, water.cp
1562.42 21.671 30.000 4628.5 5350.9 8.0000 2.7190
```

```
>>> water=IAPWS95(rho=30, s=4.)
>>> water.T, water.P, water.rho, water.u, water.h, water.s, water.x
495.00 2.4029 30.000 1597.3 1677.4 4.0000 0.39218
```

```
>>> water=IAPWS95(rho=300, u=1000.)
>>> water.T, water.P, water.rho, water.u, water.h, water.s, water.x
496.44 2.4691 300.000 1000.0 1008.2 2.6476 0.02680
```

```
>>> water=IAPWS95(s=3., h=1000.)
>>> water.T, water.P, water.rho, water.u, water.h, water.s, water.x
345.73 0.034850 0.73526 952.60 1000.0 3.0000 0.29920
```

```
>>> water=IAPWS95(u=995., h=1000.)
>>> water.T, water.P, water.rho, water.u, water.h, water.s, water.x
501.89 2.7329 546.58 995.00 1000.0 2.6298 0.00866
```

```
>>> water=IAPWS95(u=1000., s=3.)
>>> water.T, water.P, water.rho, water.u, water.h, water.s, water.x
371.24 0.094712 1.99072 1000.00 1047.6 3.0000 0.28144
```

References

IAPWS, Revised Release on the IAPWS Formulation 1995 for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use, September 2016, http://www.iapws.org/relguide/IAPWS-95.html

IAPWS, Revised Supplementary Release on Saturation Properties of Ordinary Water Substance September 1992, http://www.iapws.org/relguide/Supp-sat.html

IAPWS, Guideline on a Low-Temperature Extension of the IAPWS-95 Formulation for Water Vapor, http://www.iapws.org/relguide/LowT.html

IAPWS, Revised Advisory Note No. 3: Thermodynamic Derivatives from IAPWS Formulations, http://www.iapws.org/relguide/Advise3.pdf

Attributes

CP

Gruneisen

IntP

Ks

Kt

Prandt

Z

Z_rho

a

alfa

alfap

alfav

betap

betas

calculable Check if inputs are enough to define state

cp

cp_cv

cv

dhdP_T

dhdP_rho

dhdT_P

dhdT_rho

dhdrho_P

dhdrho_T

dpdT_rho

dpdrho_T

drhodP_T

drhodT_P

epsilon

f

fi

g

gamma

h

hInput

joule

k

kappa

ks

kt

mu

n

nu

rho

s

u

v

w

Methods

`__call__`(**kwargs)	Make instance callable to can add input parameter one to one
`calculo`()	Calculate procedure
`derivative`(z, x, y, fase)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, rho, u, h, s, g, a
`fill`(fase, estado)	Fill phase properties

name = 'water'

CASNumber = '7732-18-5'

formula = 'H2O'

synonym = 'R-718'

Tc = 647.096

rhoc = 322.0

Pc = 22.064

M = 18.015268

Tt = 273.16

Tb = 373.1243

f_acent = 0.3443

momentoDipolar = 1.855

Fi0 = {'ao_exp': [0.012436, 0.97315, 1.2795, 0.96956, 0.24873], 'ao_log': [1, 3.0063

_constants = {'A': [0.32, 0.32], 'B': [0.2, 0.2], 'C': [28, 32], 'D': [700, 800], 'R':

_Pv = {'ao': [-7.85951783, 1.84408259, -11.7866497, 22.6807411, -15.9618719, 1.801225

_rhoL = {'ao': [1.99274064, 1.09965342, -0.510839303, -1.75493479, -45.5170352, -6746

_rhoG = {'ao': [-2.0315024, -2.6830294, -5.38626492, -17.2991605, -44.7586581, -63.92

_phi0(*tau*, *delta*)
Low temperature extension of the IAPWS-95

_phiex(*T*)
Low temperature extension

classmethod _alfa_sat(*T*)
Auxiliary equation for the alfa coefficient for calculate the enthalpy along the saturation line

Parameters **T** (*float*) – Temperature, [K]

Returns **alfa** – alfa coefficient, [kJ/kg]

Return type float

References

IAPWS, Revised Supplementary Release on Saturation Properties of Ordinary Water Substance September 1992, http://www.iapws.org/relguide/Supp-sat.html, Eq.4

classmethod _phi_sat(*T*)
Auxiliary equation for the phi coefficient for calculate the entropy along the saturation line

Parameters **T** (*float*) – Temperature, [K]

Returns **phi** – phi coefficient, [kJ/kgK]

Return type float

References

IAPWS, Revised Supplementary Release on Saturation Properties of Ordinary Water Substance September 1992, http://www.iapws.org/relguide/Supp-sat.html, Eq.5

classmethod _Liquid_Enthalpy(*T*)
Auxiliary equation for the specific enthalpy for saturated liquid

Parameters **T** (*float*) – Temperature, [K]

Returns **h** – Saturated liquid enthalpy, [kJ/kg]

Return type float

References

IAPWS, Revised Supplementary Release on Saturation Properties of Ordinary Water Substance September 1992, http://www.iapws.org/relguide/Supp-sat.html, Eq.6

classmethod _Vapor_Enthalpy(*T*)
Auxiliary equation for the specific enthalpy for saturated vapor

Parameters **T** (*float*) – Temperature, [K]

Returns **h** – Saturated vapor enthalpy, [kJ/kg]

Return type float

References

IAPWS, Revised Supplementary Release on Saturation Properties of Ordinary Water Substance September 1992, http://www.iapws.org/relguide/Supp-sat.html, Eq.7

classmethod _Liquid_Entropy(*T*)
Auxiliary equation for the specific entropy for saturated liquid

Parameters **T** (*float*) – Temperature, [K]

Returns **s** – Saturated liquid entropy, [kJ/kgK]

Return type float

References

IAPWS, Revised Supplementary Release on Saturation Properties of Ordinary Water Substance September 1992, http://www.iapws.org/relguide/Supp-sat.html, Eq.8

classmethod _Vapor_Entropy(*T*)
Auxiliary equation for the specific entropy for saturated vapor

Parameters **T** (*float*) – Temperature, [K]

Returns **s** – Saturated liquid entropy, [kJ/kgK]

Return type float

References

IAPWS, Revised Supplementary Release on Saturation Properties of Ordinary Water Substance September 1992, http://www.iapws.org/relguide/Supp-sat.html, Eq.9

_visco(*rho*, *T*, *fase*)

_thermo(*rho*, *T*, *fase*)

_surface(*T*)
Generic equation for the surface tension

Parameters **T** (*float*) – Temperature, [K]

Returns σ – Surface tension, [N/m]

Return type float

Notes

Need a _surf dict in the derived class with the parameters keys: sigma: coefficient exp: exponent

class iapws.iapws95.**IAPWS95_PT**(*P*, *T*)
Derivated class for direct P and T input

Attributes

CP

Gruneisen

IntP

Ks

Kt

Prandt

Z

Z_rho

a

alfa

alfap

alfav

betap

betas

calculable Check if inputs are enough to define state

cp

cp_cv

cv

dhdP_T

dhdP_rho

dhdT_P

dhdT_rho

dhdrho_P

dhdrho_T

dpdT_rho

dpdrho_T

drhodP_T

drhodT_P

epsilon

f

fi

g

gamma

h

hInput

joule

k

kappa

ks

kt

mu

n

nu

rho

s

u

v

w

Methods

`__call__`(**kwargs)	Make instance callable to can add input parameter one to one
`calculo`()	Calculate procedure
`derivative`(z, x, y, fase)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, rho, u, h, s, g, a
`fill`(fase, estado)	Fill phase properties

class iapws.iapws95.**IAPWS95_Ph**(*P*, *h*)

Derivated class for direct P and h input

Attributes

CP

Gruneisen

IntP

Ks

Kt

Prandt

Z

Z_rho

a

alfa

alfap

alfav

betap

betas

calculable Check if inputs are enough to define state

cp

cp_cv

cv

dhdP_T

dhdP_rho

dhdT_P

dhdT_rho

dhdrho_P

dhdrho_T

dpdT_rho

dpdrho_T

drhodP_T

drhodT_P

epsilon

f

fi

g

gamma

h

hInput

joule

k

kappa

ks

kt

mu

n

nu

rho

s

u

v

w

Methods

`__call__`(**kwargs)	Make instance callable to can add input parameter one to one
`calculo`()	Calculate procedure
`derivative`(z, x, y, fase)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, rho, u, h, s, g, a
`fill`(fase, estado)	Fill phase properties

class iapws.iapws95.**IAPWS95_Ps**(*P*, *s*)

Derivated class for direct P and s input

Attributes

CP

Gruneisen

IntP

Ks

Kt

Prandt

Z

Z_rho

a

alfa

alfap

alfav

betap

betas

calculable Check if inputs are enough to define state

cp

cp_cv

cv

dhdP_T

dhdP_rho

dhdT_P

dhdT_rho

dhdrho_P

dhdrho_T

dpdT_rho

dpdrho_T

drhodP_T

drhodT_P

epsilon

f

fi

g

gamma

h

hInput

joule

k

kappa

ks

kt

mu

n

nu

rho

s

u

v

w

Methods

`__call__`(**kwargs)	Make instance callable to can add input parameter one to one
`calculo`()	Calculate procedure
`derivative`(z, x, y, fase)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, rho, u, h, s, g, a
`fill`(fase, estado)	Fill phase properties

class iapws.iapws95.**IAPWS95_Px**(*P*, *x*)

Derivated class for direct P and v input

Attributes

CP

Gruneisen

IntP

Ks

Kt

Prandt

Z

Z_rho

a

alfa

alfap

alfav

betap

betas

calculable Check if inputs are enough to define state

cp

cp_cv

cv

dhdP_T

dhdP_rho

dhdT_P

dhdT_rho

dhdrho_P

dhdrho_T

dpdT_rho

dpdrho_T

drhodP_T

drhodT_P

epsilon

f

fi

g

gamma

h

hInput

joule

k

kappa

ks

kt

mu

n

nu

rho

s

u

v

w

Methods

`__call__`(**kwargs)	Make instance callable to can add input parameter one to one
`calculo`()	Calculate procedure
`derivative`(z, x, y, fase)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, rho, u, h, s, g, a

Continued on next page

Table 6 – continued from previous page

`fill`(fase, estado)	Fill phase properties

class iapws.iapws95.**IAPWS95_Tx**(*T*, *x*)

Derivated class for direct T and x input

Attributes

CP

Gruneisen

IntP

Ks

Kt

Prandt

Z

Z_rho

a

alfa

alfap

alfav

betap

betas

calculable Check if inputs are enough to define state

cp

cp_cv

cv

dhdP_T

dhdP_rho

dhdT_P

dhdT_rho

dhdrho_P

dhdrho_T

dpdT_rho

dpdrho_T

drhodP_T

drhodT_P

epsilon

f

fi

g

gamma

h

hInput

joule

k

kappa

ks

kt

mu

n

nu

rho

s

u

v

w

Methods

`__call__`(**kwargs)	Make instance callable to can add input parameter one to one
`calculo`()	Calculate procedure
`derivative`(z, x, y, fase)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, rho, u, h, s, g, a
`fill`(fase, estado)	Fill phase properties

class iapws.iapws95.**D2O**(***kwargs*)

Implementation of IAPWS Formulation for heavy water substance, for internal procedures, see MEoS base class

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]
- **rho** (*float*) – Density, [kg/m^3]
- **v** (*float*) – Specific volume, [m^3/kg]
- **h** (*float*) – Specific enthalpy, [kJ/kg]
- **s** (*float*) – Specific entropy, [kJ/kgK]
- **u** (*float*) – Specific internal energy, [kJ/kg]
- **x** (*float*) – Vapor quality, [-]
- **l** (*float, optional*) – Wavelength of light, for refractive index, [μm]

- **rho0** (*float, optional*) – Initial value of density, to improve iteration, [kg/m^3]
- **T0** (*float, optional*) – Initial value of temperature, to improve iteration, [K]
- **x0** (*Initial value of vapor quality, necessary in bad input pair definition*) – where there are two valid solution (T-h, T-s)

Notes

- It needs two incoming properties of T, P, rho, h, s, u.
- v as a alternate input parameter to rho
- T-x, P-x, preferred input pair to specified a point in two phases region

The calculated instance has the following properties:

- P: Pressure, [MPa]
- T: Temperature, [K]
- x: Vapor quality, [-]
- g: Specific Gibbs free energy, [kJ/kg]
- a: Specific Helmholtz free energy, [kJ/kg]
- v: Specific volume, [m^3/kg]
- r: Density, [kg/m^3]
- h: Specific enthalpy, [kJ/kg]
- u: Specific internal energy, [kJ/kg]
- s: Specific entropy, [kJ/kg·K]
- cp: Specific isobaric heat capacity, [kJ/kg·K]
- cv: Specific isochoric heat capacity, [kJ/kg·K]
- cp_cv: Heat capacity ratio, [-]
- Z: Compression factor, [-]
- fi: Fugacity coefficient, [-]
- f: Fugacity, [MPa]
- gamma: Isoentropic exponent, [-]
- alfav: Isobaric cubic expansion coefficient, [1/K]
- kappa: Isothermal compressibility, [1/MPa]
- kappas: Adiabatic compresibility, [1/MPa]
- alfap: Relative pressure coefficient, [1/K]
- betap: Isothermal stress coefficient, [kg/m^3]
- joule: Joule-Thomson coefficient, [K/MPa]
- betas: Isoentropic temperature-pressure coefficient, [-]
- Gruneisen: Gruneisen parameter, [-]
- virialB: Second virial coefficient, [m^3/kg]

- virialC: Third virial coefficient, [m^6/kg^2]
- dpdT_rho: Derivatives, dp/dT at constant rho, [MPa/K]
- dpdrho_T: Derivatives, dp/drho at constant T, [MPa·m^3/kg]
- drhodT_P: Derivatives, drho/dT at constant P, [kg/m^3·K]
- drhodP_T: Derivatives, drho/dP at constant T, [kg/m^3·MPa]
- dhdT_rho: Derivatives, dh/dT at constant rho, [kJ/kg·K]
- dhdP_T: Isothermal throttling coefficient, [kJ/kg·MPa]
- dhdT_P: Derivatives, dh/dT at constant P, [kJ/kg·K]
- dhdrho_T: Derivatives, dh/drho at constant T, [kJ·m^3/kg^2]
- dhdrho_P: Derivatives, dh/drho at constant P, [kJ·m^3/kg^2]
- dhdP_rho: Derivatives, dh/dP at constant rho, [kJ/kg·MPa]
- kt: Isothermal Expansion Coefficient, [-]
- ks: Adiabatic Compressibility, [1/MPa]
- Ks: Adiabatic bulk modulus, [MPa]
- Kt: Isothermal bulk modulus, [MPa]
- v0: Ideal specific volume, [m^3/kg]
- rho0: Ideal gas density, [kg/m^3]
- u0: Ideal specific internal energy, [kJ/kg]
- h0: Ideal specific enthalpy, [kJ/kg]
- s0: Ideal specific entropy, [kJ/kg·K]
- a0: Ideal specific Helmholtz free energy, [kJ/kg]
- g0: Ideal specific Gibbs free energy, [kJ/kg]
- cp0: Ideal specific isobaric heat capacity, [kJ/kg·K]
- cv0: Ideal specific isochoric heat capacity, [kJ/kg·K]
- w0: Ideal speed of sound, [m/s]
- gamma0: Ideal isoentropic exponent, [-]
- w: Speed of sound, [m/s]
- mu: Dynamic viscosity, [Pa·s]
- nu: Kinematic viscosity, [m^2/s]
- k: Thermal conductivity, [W/m·K]
- alfa: Thermal diffusivity, [m^2/s]
- sigma: Surface tension, [N/m]
- epsilon: Dielectric constant, [-]
- n: Refractive index, [-]
- Prandt: Prandtl number, [-]
- Pr: Reduced Pressure, [-]

- Tr: Reduced Temperature, [-]
- Hvap: Vaporization heat, [kJ/kg]
- Svap: Vaporization entropy, [kJ/kg·K]
- Z_rho: $(Z-1)/\rho$, [m^3/kg]
- IntP: Internal pressure, [MPa]
- invT: Negative reciprocal temperature, [1/K]
- hInput: Specific heat input, [kJ/kg]

Examples

```
>>> hwater=D2O(T=300, rho=996.5560)
>>> hwater.P, hwater.Liquid.cv, hwater.Liquid.w
0.0030675947 4.21191157 5332.04871
```

References

IAPWS, Release on the IAPWS Formulation 2017 for the Thermodynamic Properties of Heavy Water, http://www.iapws.org/relguide/Heavy-2017.pdf IAPWS, Revised Advisory Note No. 3: Thermodynamic Derivatives from IAPWS Formulations, http://www.iapws.org/relguide/Advise3.pdf

Attributes

CP

Gruneisen

IntP

Ks

Kt

Prandt

Z

Z_rho

a

alfa

alfap

alfav

betap

betas

`calculable` Check if inputs are enough to define state

cp

cp_cv

cv

dhdP_T

dhdP_rho

dhdT_P

dhdT_rho

dhdrho_P

dhdrho_T

dpdT_rho

dpdrho_T

drhodP_T

drhodT_P

epsilon

f

fi

g

gamma

h

hInput

joule

k

kappa

ks

kt

mu

n

nu

rho

s

u

v

w

Methods

`__call__`(**kwargs)	Make instance callable to can add input parameter one to one
`calculo`()	Calculate procedure
`derivative`(z, x, y, fase)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, rho, u, h, s, g, a

Continued on next page

Table 8 – continued from previous page

`fill`(fase, estado)	Fill phase properties

```
name = 'heavy water'
CASNumber = '7789-20-0'
formula = 'D2O'
synonym = 'deuterium oxide'
Tc = 643.847
rhoc = 355.9999698294
Pc = 21.6618
M = 20.027508
Tt = 276.97
Tb = 374.563
f_acent = 0.364
momentoDipolar = 1.9
Fi0 = {'ao_exp':  [0.010633, 0.99787, 2.1483, 0.3549], 'ao_hyp':  [], 'ao_log':  [1, 3
_constants = {'R': 8.3144598, 'alfa3':  [0.6014, 1.4723, 1.5305, 2.4297, 1.3086, 1.352
_Pv = {'ao':  [-8.0236, 2.3957, -42.639, 99.569, -62.135], 'exp':  [1.0, 1.5, 2.75, 3.
_rhoL = {'ao':  [2.6406, 9.709, -18.058, 8.7202, -7.4487], 'eq':  1, 'exp':  [0.3678,
_rhoG = {'ao':  [-3.7651, -38.673, 73.024, -132.51, 75.235, -70.412], 'eq':  3, 'exp':
```

_visco(*rho*, *T*, *fase*)

_thermo(*rho*, *T*, *fase*)

_surface(*T*)

Generic equation for the surface tension

Parameters **T** (*float*) – Temperature, [K]

Returns σ – Surface tension, [N/m]

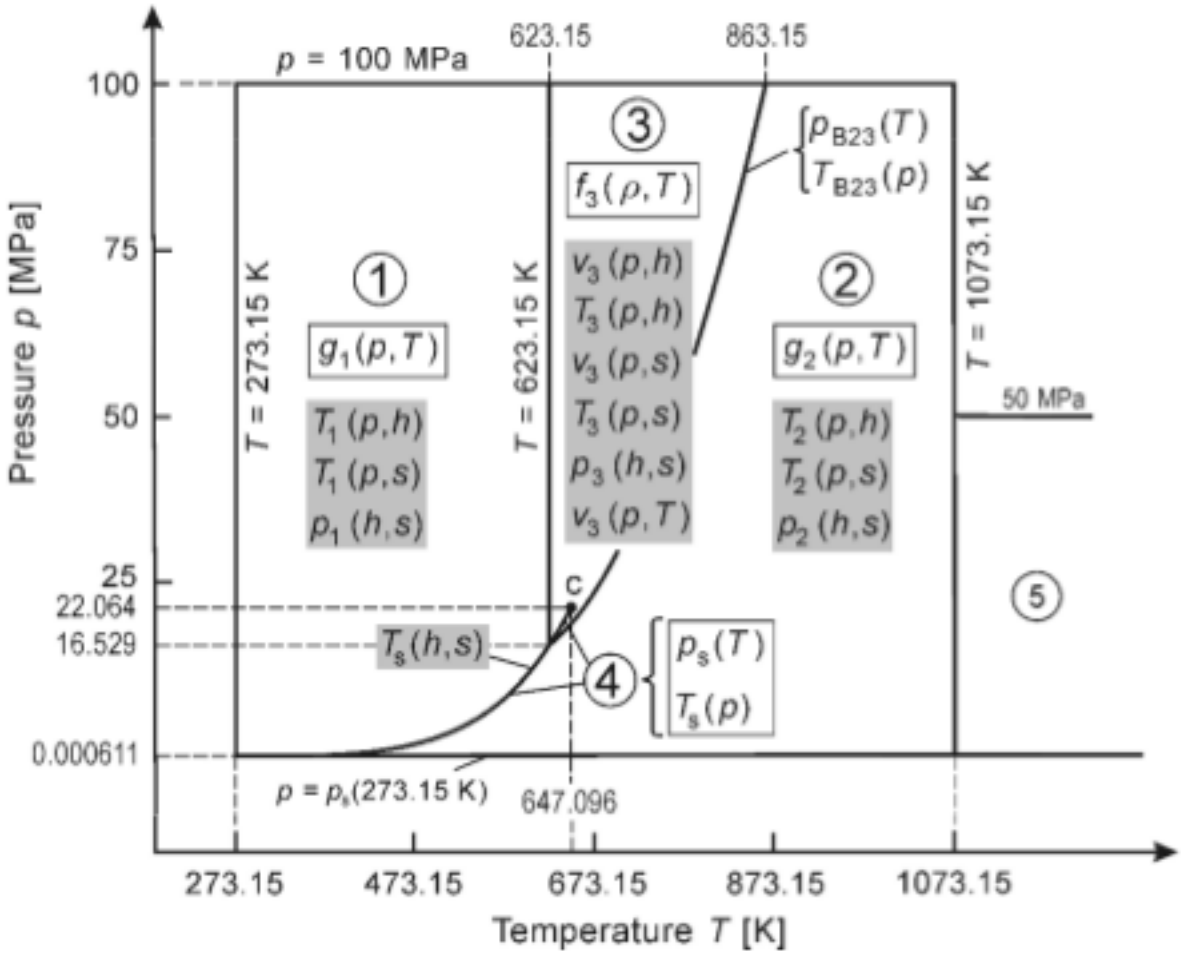
Return type float

Notes

Need a _surf dict in the derived class with the parameters keys: sigma: coefficient exp: exponent

5.1.4 iapws.iapws97 module

IAPWS-IF97 standard implementation



The module implement the fundamental equation for the five regions (rectangular boxes) and the backward equation (marked in grey).

IAPWS97: Global module class with all the functionality integrated

Fundamental equations:

- *_Region1()*
- *_Region2()*
- *_Region3()*
- *_Region4()*
- *_TSat_P()*
- *_PSat_T()*
- *_Region5()*

Backward equations:

- *_Backward1_T_Ph()*
- *_Backward1_T_Ps()*
- *_Backward1_P_hs()*
- *_Backward2_T_Ph()*

- *_Backward2_T_Ps()*
- *_Backward2_P_hs()*
- *_Backward3_T_Ph()*
- *_Backward3_T_Ps()*
- *_Backward3_P_hs()*
- *_Backward3_v_Ph()*
- *_Backward3_v_Ps()*
- *_Backward3_v_PT()*
- *_Backward4_T_hs()*

Boundary equations:

- *_h13_s()*
- *_h3a_s()*
- *_h1_s()*
- *_t_hs()*
- *_PSat_h()*
- *_h2ab_s()*
- *_h_3ab()*
- *_h2c3b_s()*
- *_hab_s()*
- *_hbc_P()*

References:

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html

IAPWS, Revised Supplementary Release on Backward Equations for Pressure as a Function of Enthalpy and Entropy p(h,s) for Regions 1 and 2 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-PHS12-2014.pdf

IAPWS, Revised Supplementary Release on Backward Equations for the Functions T(p,h), v(p,h) and T(p,s), v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-Tv%28ph,ps%293-2014.pdf

IAPWS, Revised Supplementary Release on Backward Equations p(h,s) for Region 3, Equations as a Function of h and s for the Region Boundaries, and an Equation Tsat(h,s) for Region 4 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-phs3-2014.pdf

IAPWS, Revised Supplementary Release on Backward Equations for Specific Volume as a Function of Pressure and Temperature v(p,T) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-VPT3-2016.pdf

IAPWS, Revised Advisory Note No. 3: Thermodynamic Derivatives from IAPWS Formulations, http://www.iapws.org/relguide/Advise3.pdf

Wagner, W; Kretzschmar, H-J: International Steam Tables: Properties of Water and Steam Based on the Industrial Formulation IAPWS-IF97; Springer, 2008; doi: 10.1007/978-3-540-74234-0

iapws.iapws97.**_h13_s**(*s*)

Parameters **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **h** – Specific enthalpy, [kJ/kg]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- s(100MPa,623.15K) s s'(623.15K)

References

IAPWS, Revised Supplementary Release on Backward Equations p(h,s) for Region 3, Equations as a Function of h and s for the Region Boundaries, and an Equation Tsat(h,s) for Region 4 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-phs3-2014.pdf. Eq 7

Examples

```
>>> _h13_s(3.7)
1632.525047
>>> _h13_s(3.5)
1566.104611
```

iapws.iapws97.**_P23_T**(*T*)

Parameters **T** (*float*) – Temperature, [K]

Returns **P** – Pressure, [MPa]

Return type float

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 5

Examples

```
>>> _P23_T(623.15)
16.52916425
```

iapws.iapws97.**_t_P**(*P*)

Parameters **P** (*float*) – Pressure, [MPa]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 5

Examples

```
>>> _t_P(16.52916425)
623.15
```

iapws.iapws97.**_t_hs**(*h*, *s*)

Parameters

- **h** (*float*) – Specific enthalpy, [kJ/kg]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **T** – Temperature, [K]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- 5.048096828 ≤ s ≤ 5.260578707
- 2.563592004e3 ≤ h ≤ 2.812942061e3

References

IAPWS, Revised Supplementary Release on Backward Equations p(h,s) for Region 3, Equations as a Function of h and s for the Region Boundaries, and an Equation Tsat(h,s) for Region 4 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-phs3-2014.pdf. Eq 8

Examples

```
>>> _t_hs(2600, 5.1)
713.5259364
>>> _t_hs(2800, 5.2)
817.6202120
```

iapws.iapws97.**_PSat_T**(*T*)

Parameters **T** (*float*) – Temperature, [K]

Returns **P** – Pressure, [MPa]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- 273.15 T 647.096

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 30

Examples

```
>>> _PSat_T(500)
2.63889776
```

`iapws.iapws97._TSat_P(P)`

Parameters **P** (*float*) – Pressure, [MPa]

Returns **T** – Temperature, [K]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- 0.00061121 P 22.064

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 31

Examples

```
>>> _TSat_P(10)
584.149488
```

`iapws.iapws97._PSat_h(h)`

Define the saturated line, P=f(h) for region 3

Parameters **h** (*float*) – Specific enthalpy, [kJ/kg]

Returns **P** – Pressure, [MPa]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- h'(623.15K) h h"(623.15K)

References

IAPWS, Revised Supplementary Release on Backward Equations for the Functions T(p,h), v(p,h) and T(p,s), v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-Tv%28ph,ps%293-2014.pdf, Eq 10

Examples

```
>>> _PSat_h(1700)
17.24175718
>>> _PSat_h(2400)
20.18090839
```

iapws.iapws97.**_PSat_s**(*s*)

Define the saturated line, P=f(s) for region 3

Parameters **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **P** – Pressure, [MPa]

Return type float

Notes

Raise NotImplementedError if input isn't in limit:

- s'(623.15K) ≤ s ≤ s"(623.15K)

References

IAPWS, Revised Supplementary Release on Backward Equations for the Functions T(p,h), v(p,h) and T(p,s), v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-Tv%28ph,ps%293-2014.pdf, Eq 11

Examples

```
>>> _PSat_s(3.8)
16.87755057
>>> _PSat_s(5.2)
16.68968482
```

iapws.iapws97.**_h1_s**(*s*)

Parameters **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **h** – Specific enthalpy, [kJ/kg]

Return type float

Notes

Raise NotImplementedError if input isn't in limit:

- s'(273.15K) ≤ s ≤ s'(623.15K)

References

IAPWS, Revised Supplementary Release on Backward Equations p(h,s) for Region 3, Equations as a Function of h and s for the Region Boundaries, and an Equation Tsat(h,s) for Region 4 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-phs3-2014.pdf. Eq 3

Examples

```
>>> _h1_s(1)
308.5509647
>>> _h1_s(3)
1198.359754
```

iapws.iapws97.**_h3a_s**(*s*)

Parameters **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **h** – Specific enthalpy, [kJ/kg]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- s'(623.15K) s sc

References

IAPWS, Revised Supplementary Release on Backward Equations p(h,s) for Region 3, Equations as a Function of h and s for the Region Boundaries, and an Equation Tsat(h,s) for Region 4 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-phs3-2014.pdf. Eq 4

Examples

```
>>> _h3a_s(3.8)
1685.025565
>>> _h3a_s(4.2)
1949.352563
```

iapws.iapws97.**_h2ab_s**(*s*)

Define the saturated line boundary between Region 4 and 2a-2b, h=f(s)

Parameters **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **h** – Specific enthalpy, [kJ/kg]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- 5.85 ≤ s ≤ s"(273.15K)

References

IAPWS, Revised Supplementary Release on Backward Equations p(h,s) for Region 3, Equations as a Function of h and s for the Region Boundaries, and an Equation Tsat(h,s) for Region 4 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-phs3-2014.pdf. Eq 5

Examples

```
>>> _h2ab_s(7)
2723.729985
>>> _h2ab_s(9)
2511.861477
```

iapws.iapws97.**_h2c3b_s**(*s*)

Define the saturated line boundary between Region 4 and 2c-3b, h=f(s)

Parameters **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **h** – Specific enthalpy, [kJ/kg]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- sc ≤ s ≤ 5.85

References

IAPWS, Revised Supplementary Release on Backward Equations p(h,s) for Region 3, Equations as a Function of h and s for the Region Boundaries, and an Equation Tsat(h,s) for Region 4 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-phs3-2014.pdf. Eq 6

Examples

```
>>> _h2c3b_s(5.5)
2687.693850
>>> _h2c3b_s(4.5)
2144.360448
```

iapws.iapws97.**_Region1**(*T*, *P*)

Basic equation for region 1

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]

Returns

prop –

Dict with calculated properties. The available properties are:

- v: Specific volume, [m^3/kg]
- h: Specific enthalpy, [kJ/kg]
- s: Specific entropy, [kJ/kgK]
- cp: Specific isobaric heat capacity, [kJ/kgK]
- cv: Specific isocoric heat capacity, [kJ/kgK]
- w: Speed of sound, [m/s]
- alfav: Cubic expansion coefficient, [1/K]
- kt: Isothermal compressibility, [1/MPa]

Return type dict

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 7

Examples

```
>>> _Region1(300,3)["v"]
0.00100215168
>>> _Region1(300,3)["h"]
115.331273
>>> _Region1(300,3)["h"]-3000*_Region1(300,3)["v"]
112.324818
>>> _Region1(300,80)["s"]
0.368563852
>>> _Region1(300,80)["cp"]
4.01008987
>>> _Region1(300,80)["cv"]
3.91736606
>>> _Region1(500,3)["w"]
1240.71337
>>> _Region1(500,3)["alfav"]
0.00164118128
>>> _Region1(500,3)["kt"]
0.00112892188
```

iapws.iapws97.**_Backward1_T_Ph**(*P*, *h*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **h** (*float*) – Specific enthalpy, [kJ/kg]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 11

Examples

```
>>> _Backward1_T_Ph(3,500)
391.798509
>>> _Backward1_T_Ph(80,1500)
611.041229
```

iapws.iapws97.**_Backward1_T_Ps**(*P*, *s*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 13

Examples

```
>>> _Backward1_T_Ps(3,0.5)
307.842258
>>> _Backward1_T_Ps(80,3)
565.899909
```

iapws.iapws97.**_Backward1_P_hs**(*h*, *s*)

Parameters

- **h** (*float*) – Specific enthalpy, [kJ/kg]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **P** – Pressure, [MPa]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for Pressure as a Function of Enthalpy and Entropy p(h,s) for Regions 1 and 2 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-PHS12-2014.pdf, Eq 1

Examples

```
>>> _Backward1_P_hs(0.001,0)
0.0009800980612
>>> _Backward1_P_hs(90,0)
91.92954727
>>> _Backward1_P_hs(1500,3.4)
58.68294423
```

iapws.iapws97.**_Region2**(*T*, *P*)

Basic equation for region 2

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]

Returns

prop –

Dict with calculated properties. The available properties are:

- v: Specific volume, [m^3/kg]
- h: Specific enthalpy, [kJ/kg]
- s: Specific entropy, [kJ/kgK]
- cp: Specific isobaric heat capacity, [kJ/kgK]
- cv: Specific isocoric heat capacity, [kJ/kgK]
- w: Speed of sound, [m/s]
- alfav: Cubic expansion coefficient, [1/K]
- kt: Isothermal compressibility, [1/MPa]

Return type dict

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 15-17

Examples

```
>>> _Region2(700,30)["v"]
0.00542946619
>>> _Region2(700,30)["h"]
2631.49474
```

(continues on next page)

(continued from previous page)

```
>>> _Region2(700,30)["h"]-30000*_Region2(700,30)["v"]
2468.61076
>>> _Region2(700,0.0035)["s"]
10.1749996
>>> _Region2(700,0.0035)["cp"]
2.08141274
>>> _Region2(700,0.0035)["cv"]
1.61978333
>>> _Region2(300,0.0035)["w"]
427.920172
>>> _Region2(300,0.0035)["alfav"]
0.00337578289
>>> _Region2(300,0.0035)["kt"]
286.239651
```

iapws.iapws97.**Region2_cp0**(*Tr*, *Pr*)

Ideal properties for Region 2

Parameters

- **Tr** (*float*) – Reduced temperature, [-]
- **Pr** (*float*) – Reduced pressure, [-]

Returns

prop –

Array with ideal Gibbs energy partial derivatives:

- g: Ideal Specific Gibbs energy [kJ/kg]
- gp: g/P|T
- gpp: ^{2}g/P^2|T
- gt: g/T|P
- gtt: ^{2}g/T^2|P
- gpt: ^{2}g/TP

Return type array

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 16

iapws.iapws97.**_P_2bc**(*h*)

Parameters **h** (*float*) – Specific enthalpy, [kJ/kg]

Returns **P** – Pressure, [MPa]

Return type float

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 20

Examples

```
>>> _P_2bc(3516.004323)
100.0
```

iapws.iapws97.**_hbc_P**(*P*)

Parameters **P** (*float*) – Pressure, [MPa]

Returns **h** – Specific enthalpy, [kJ/kg]

Return type float

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 21

Examples

```
>>> _hbc_P(100)
3516.004323
```

iapws.iapws97.**_hab_s**(*s*)

Parameters **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **h** – Specific enthalpy, [kJ/kg]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for Pressure as a Function of Enthalpy and Entropy p(h,s) for Regions 1 and 2 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-PHS12-2014.pdf, Eq 2

Examples

```
>>> _hab_s(7)
3376.437884
```

iapws.iapws97.**_Backward2a_T_Ph**(*P*, *h*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **h** (*float*) – Specific enthalpy, [kJ/kg]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 22

Examples

```
>>> _Backward2a_T_Ph(0.001,3000)
534.433241
>>> _Backward2a_T_Ph(3,4000)
1010.77577
```

iapws.iapws97.**_Backward2b_T_Ph**(*P*, *h*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **h** (*float*) – Specific enthalpy, [kJ/kg]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 23

Examples

```
>>> _Backward2b_T_Ph(5,4000)
1015.31583
>>> _Backward2b_T_Ph(25,3500)
875.279054
```

iapws.iapws97.**_Backward2c_T_Ph**(*P*, *h*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **h** (*float*) – Specific enthalpy, [kJ/kg]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 24

Examples

```
>>> _Backward2c_T_Ph(40,2700)
743.056411
>>> _Backward2c_T_Ph(60,3200)
882.756860
```

iapws.iapws97.**_Backward2_T_Ph**(*P*, *h*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **h** (*float*) – Specific enthalpy, [kJ/kg]

Returns **T** – Temperature, [K]

Return type float

iapws.iapws97.**_Backward2a_T_Ps**(*P*, *s*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 25

Examples

```
>>> _Backward2a_T_Ps(0.1,7.5)
399.517097
>>> _Backward2a_T_Ps(2.5,8)
1039.84917
```

iapws.iapws97.**_Backward2b_T_Ps**(*P*, *s*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 26

Examples

```
>>> _Backward2b_T_Ps(8,6)
600.484040
>>> _Backward2b_T_Ps(90,6)
1038.01126
```

iapws.iapws97.**_Backward2c_T_Ps**(*P*, *s*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 27

Examples

```
>>> _Backward2c_T_Ps(20,5.75)
697.992849
>>> _Backward2c_T_Ps(80,5.75)
949.017998
```

iapws.iapws97.**_Backward2_T_Ps**(*P*, *s*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **T** – Temperature, [K]

Return type float

iapws.iapws97.**_Backward2a_P_hs**(*h*, *s*)

Parameters

- **h** (*float*) – Specific enthalpy, [kJ/kg]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **P** – Pressure, [MPa]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for Pressure as a Function of Enthalpy and Entropy p(h,s) for Regions 1 and 2 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-PHS12-2014.pdf, Eq 3

Examples

```
>>> _Backward2a_P_hs(2800,6.5)
1.371012767
>>> _Backward2a_P_hs(2800,9.5)
0.001879743844
>>> _Backward2a_P_hs(4100,9.5)
0.1024788997
```

iapws.iapws97.**_Backward2b_P_hs**(*h*, *s*)

Parameters

- **h** (*float*) – Specific enthalpy, [kJ/kg]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **P** – Pressure, [MPa]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for Pressure as a Function of Enthalpy and Entropy p(h,s) for Regions 1 and 2 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-PHS12-2014.pdf, Eq 4

Examples

```
>>> _Backward2b_P_hs(2800,6)
4.793911442
>>> _Backward2b_P_hs(3600,6)
83.95519209
>>> _Backward2b_P_hs(3600,7)
7.527161441
```

iapws.iapws97.**_Backward2c_P_hs**(*h*, *s*)

Parameters

- **h** (*float*) – Specific enthalpy, [kJ/kg]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **P** – Pressure, [MPa]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for Pressure as a Function of Enthalpy and Entropy p(h,s) for Regions 1 and 2 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-PHS12-2014.pdf, Eq 5

Examples

```
>>> _Backward2c_P_hs(2800,5.1)
94.39202060
>>> _Backward2c_P_hs(2800,5.8)
8.414574124
>>> _Backward2c_P_hs(3400,5.8)
83.76903879
```

iapws.iapws97.**_Backward2_P_hs**(*h*, *s*)

Parameters

- **h** (*float*) – Specific enthalpy, [kJ/kg]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **P** – Pressure, [MPa]

Return type float

iapws.iapws97.**_Region3**(*rho*, *T*)

Basic equation for region 3

Parameters

- **rho** (*float*) – Density, [kg/m^3]
- **T** (*float*) – Temperature, [K]

Returns

prop –

Dict with calculated properties. The available properties are:

- v: Specific volume, [m^3/kg]
- h: Specific enthalpy, [kJ/kg]
- s: Specific entropy, [kJ/kgK]
- cp: Specific isobaric heat capacity, [kJ/kgK]
- cv: Specific isocoric heat capacity, [kJ/kgK]
- w: Speed of sound, [m/s]
- alfav: Cubic expansion coefficient, [1/K]
- kt: Isothermal compressibility, [1/MPa]

Return type dict

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 28

Examples

```
>>> _Region3(500,650)["P"]
25.5837018
>>> _Region3(500,650)["h"]
1863.43019
>>> p = _Region3(500, 650)
>>> p["h"]-p["P"]*1000*p["v"]
1812.26279
>>> _Region3(200,650)["s"]
4.85438792
>>> _Region3(200,650)["cp"]
44.6579342
>>> _Region3(200,650)["cv"]
4.04118076
>>> _Region3(200,650)["w"]
383.444594
>>> _Region3(500,750)["alfav"]
0.00441515098
>>> _Region3(500,750)["kt"]
0.00806710817
```

iapws.iapws97.**_h_3ab**(*P*)

Define the boundary between Region 3a-3b, h=f(P)

Parameters **P** (*float*) – Pressure, [MPa]

Returns **h** – Specific enthalpy, [kJ/kg]

Return type float

Examples

```
>>> _h_3ab(25)
2095.936454
```

iapws.iapws97.**_tab_P**(*P*)

Define the boundary between Region 3a-3b, T=f(P)

Parameters **P** (*float*) – Pressure, [MPa]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for Specific Volume as a Function of Pressure and Temperature v(p,T) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-VPT3-2016.pdf, Eq. 2

Examples

```
>>> _tab_P(40)
693.0341408
```

iapws.iapws97.**_top_P**(*P*)

Define the boundary between Region 3o-3p, T=f(P)

Parameters **P** (*float*) – Pressure, [MPa]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for Specific Volume as a Function of Pressure and Temperature v(p,T) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-VPT3-2016.pdf, Eq. 2

Examples

```
>>> _top_P(22.8)
650.0106943
```

iapws.iapws97.**_twx_P**(*P*)

Define the boundary between Region 3w-3x, T=f(P)

Parameters **P** (*float*) – Pressure, [MPa]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for Specific Volume as a Function of Pressure and Temperature v(p,T) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-VPT3-2016.pdf, Eq. 2

Examples

```
>>> _twx_P(22.3)
648.2049480
```

iapws.iapws97.**_tef_P**(*P*)

Define the boundary between Region 3e-3f, T=f(P)

Parameters **P** (*float*) – Pressure, [MPa]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for Specific Volume as a Function of Pressure and Temperature v(p,T) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-VPT3-2016.pdf, Eq. 3

Examples

```
>>> _tef_P(40)
713.9593992
```

iapws.iapws97.**_txx_P**(*P*, *xy*)

Define the boundary between 3x-3y, T=f(P)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **xy** (*string*) – Subregions options: cd, gh, ij, jk, mn, qu, rx, uv

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for Specific Volume as a Function of Pressure and Temperature v(p,T) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-VPT3-2016.pdf, Eq. 1

Examples

```
>>> _txx_P(25,"cd")
649.3659208
>>> _txx_P(23,"gh")
649.8873759
>>> _txx_P(23,"ij")
651.5778091
>>> _txx_P(23,"jk")
655.8338344
>>> _txx_P(22.8,"mn")
649.6054133
>>> _txx_P(22,"qu")
645.6355027
>>> _txx_P(22,"rx")
648.2622754
>>> _txx_P(22.3,"uv")
647.7996121
```

iapws.iapws97.**_Backward3a_v_Ph**(*P*, *h*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **h** (*float*) – Specific enthalpy, [kJ/kg]

References

IAPWS, Revised Supplementary Release on Backward Equations for the Functions T(p,h), v(p,h) and T(p,s), v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-Tv%28ph,ps%293-2014.pdf, Eq 4

Returns **v** – Specific volume, [m^3/kg]

Return type float

Examples

```
>>> _Backward3a_v_Ph(20,1700)
0.001749903962
>>> _Backward3a_v_Ph(100,2100)
0.001676229776
```

iapws.iapws97.**_Backward3b_v_Ph**(*P*, *h*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **h** (*float*) – Specific enthalpy, [kJ/kg]

Returns **v** – Specific volume, [m^3/kg]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for the Functions T(p,h), v(p,h) and T(p,s), v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-Tv%28ph,ps%293-2014.pdf, Eq 5

Examples

```
>>> _Backward3b_v_Ph(20,2500)
0.006670547043
>>> _Backward3b_v_Ph(100,2700)
0.002404234998
```

iapws.iapws97.**_Backward3_v_Ph**(*P*, *h*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **h** (*float*) – Specific enthalpy, [kJ/kg]

Returns **v** – Specific volume, [m^3/kg]

Return type float

iapws.iapws97.**_Backward3a_T_Ph**(*P*, *h*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **h** (*float*) – Specific enthalpy, [kJ/kg]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for the Functions T(p,h), v(p,h) and T(p,s), v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-Tv%28ph,ps%293-2014.pdf, Eq 2

Examples

```
>>> _Backward3a_T_Ph(20,1700)
629.3083892
>>> _Backward3a_T_Ph(100,2100)
733.6163014
```

iapws.iapws97.**_Backward3b_T_Ph**(*P*, *h*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **h** (*float*) – Specific enthalpy, [kJ/kg]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for the Functions T(p,h), v(p,h) and T(p,s), v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-Tv%28ph,ps%293-2014.pdf, Eq 3

Examples

```
>>> _Backward3b_T_Ph(20,2500)
641.8418053
>>> _Backward3b_T_Ph(100,2700)
842.0460876
```

iapws.iapws97.**_Backward3_T_Ph**(*P*, *h*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **h** (*float*) – Specific enthalpy, [kJ/kg]

Returns **T** – Temperature, [K]

Return type float

iapws.iapws97.**_Backward3a_v_Ps**(*P*, *s*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **v** – Specific volume, [m^3/kg]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for the Functions T(p,h), v(p,h) and T(p,s), v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-Tv%28ph,ps%293-2014.pdf, Eq 8

Examples

```
>>> _Backward3a_v_Ps(20,3.8)
0.001733791463
>>> _Backward3a_v_Ps(100,4)
0.001555893131
```

iapws.iapws97.**_Backward3b_v_Ps**(*P*, *s*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **v** – Specific volume, [m^3/kg]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for the Functions T(p,h), v(p,h) and T(p,s), v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-Tv%28ph,ps%293-2014.pdf, Eq 9

Examples

```
>>> _Backward3b_v_Ps(20,5)
0.006262101987
>>> _Backward3b_v_Ps(100,5)
0.002449610757
```

iapws.iapws97.**_Backward3_v_Ps**(*P*, *s*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **v** – Specific volume, [m^3/kg]

Return type float

iapws.iapws97.**_Backward3a_T_Ps**(*P*, *s*)

Parameters

- **P** (*float*) – Pressure, [MPa]

- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for the Functions T(p,h), v(p,h) and T(p,s), v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-Tv%28ph,ps%293-2014.pdf, Eq 6

Examples

```
>>> _Backward3a_T_Ps(20,3.8)
628.2959869
>>> _Backward3a_T_Ps(100,4)
705.6880237
```

iapws.iapws97.**_Backward3b_T_Ps**(*P*, *s*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for the Functions T(p,h), v(p,h) and T(p,s), v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-Tv%28ph,ps%293-2014.pdf, Eq 7

Examples

```
>>> _Backward3b_T_Ps(20,5)
640.1176443
>>> _Backward3b_T_Ps(100,5)
847.4332825
```

iapws.iapws97.**_Backward3_T_Ps**(*P*, *s*)

Parameters

- **P** (*float*) – Pressure, [MPa]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **T** – Temperature, [K]

Return type float

iapws.iapws97.**_Backward3a_P_hs**(*h*, *s*)

Parameters

- **h** (*float*) – Specific enthalpy, [kJ/kg]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **P** – Pressure, [MPa]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations p(h,s) for Region 3, Equations as a Function of h and s for the Region Boundaries, and an Equation Tsat(h,s) for Region 4 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-phs3-2014.pdf. Eq 1

Examples

```
>>> _Backward3a_P_hs(1700,3.8)
25.55703246
>>> _Backward3a_P_hs(2000,4.2)
45.40873468
>>> _Backward3a_P_hs(2100,4.3)
60.78123340
```

iapws.iapws97.**_Backward3b_P_hs**(*h*, *s*)

Parameters

- **h** (*float*) – Specific enthalpy, [kJ/kg]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **P** – Pressure, [MPa]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations p(h,s) for Region 3, Equations as a Function of h and s for the Region Boundaries, and an Equation Tsat(h,s) for Region 4 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-phs3-2014.pdf. Eq 1

Examples

```
>>> _Backward3b_P_hs(2400,4.7)
63.63924887
>>> _Backward3b_P_hs(2600,5.1)
34.34999263
>>> _Backward3b_P_hs(2700,5.0)
88.39043281
```

iapws.iapws97.**_Backward3_P_hs**(*h*, *s*)

Parameters

- **h** (*float*) – Specific enthalpy, [kJ/kg]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **P** – Pressure, [MPa]

Return type float

iapws.iapws97.**_Backward3_sat_v_P**(*P*, *x*)

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]
- **x** (*integer*) – Vapor quality, [-]

Returns **v** – Specific volume, [m^3/kg]

Return type float

Notes

The vapor quality (x) can be 0 (saturated liquid) or 1 (saturated vapour)

iapws.iapws97.**_Backward3_v_PT**(*P*, *T*)

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]

Returns **v** – Specific volume, [m^3/kg]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for Specific Volume as a Function of Pressure and Temperature v(p,T) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-VPT3-2016.pdf, Table 2 and 10

iapws.iapws97.**_Backward3x_v_PT**(*P*, *T*)

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]
- **x** (*char*) – Region 3 subregion code

Returns **v** – Specific volume, [m^3/kg]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations for Specific Volume as a Function of Pressure and Temperature v(p,T) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-VPT3-2016.pdf, Eq. 4-5

Examples

```
>>> _Backward3x_v_PT(630,50,"a")
0.001470853100
>>> _Backward3x_v_PT(670,80,"a")
0.001503831359
>>> _Backward3x_v_PT(710,50,"b")
0.002204728587
>>> _Backward3x_v_PT(750,80,"b")
0.001973692940
>>> _Backward3x_v_PT(630,20,"c")
0.001761696406
>>> _Backward3x_v_PT(650,30,"c")
0.001819560617
>>> _Backward3x_v_PT(656,26,"d")
0.002245587720
>>> _Backward3x_v_PT(670,30,"d")
0.002506897702
>>> _Backward3x_v_PT(661,26,"e")
0.002970225962
>>> _Backward3x_v_PT(675,30,"e")
0.003004627086
>>> _Backward3x_v_PT(671,26,"f")
0.005019029401
>>> _Backward3x_v_PT(690,30,"f")
0.004656470142
>>> _Backward3x_v_PT(649,23.6,"g")
0.002163198378
>>> _Backward3x_v_PT(650,24,"g")
0.002166044161
>>> _Backward3x_v_PT(652,23.6,"h")
0.002651081407
>>> _Backward3x_v_PT(654,24,"h")
0.002967802335
>>> _Backward3x_v_PT(653,23.6,"i")
0.003273916816
>>> _Backward3x_v_PT(655,24,"i")
0.003550329864
>>> _Backward3x_v_PT(655,23.5,"j")
0.004545001142
>>> _Backward3x_v_PT(660,24,"j")
0.005100267704
>>> _Backward3x_v_PT(660,23,"k")
0.006109525997
>>> _Backward3x_v_PT(670,24,"k")
0.006427325645
>>> _Backward3x_v_PT(646,22.6,"l")
0.002117860851
>>> _Backward3x_v_PT(646,23,"l")
0.002062374674
>>> _Backward3x_v_PT(648.6,22.6,"m")
0.002533063780
>>> _Backward3x_v_PT(649.3,22.8,"m")
0.002572971781
>>> _Backward3x_v_PT(649,22.6,"n")
0.002923432711
>>> _Backward3x_v_PT(649.7,22.8,"n")
```

(continues on next page)

(continued from previous page)

```
0.002913311494
>>> _Backward3x_v_PT(649.1,22.6,"o")
0.003131208996
>>> _Backward3x_v_PT(649.9,22.8,"o")
0.003221160278
>>> _Backward3x_v_PT(649.4,22.6,"p")
0.003715596186
>>> _Backward3x_v_PT(650.2,22.8,"p")
0.003664754790
>>> _Backward3x_v_PT(640,21.1,"q")
0.001970999272
>>> _Backward3x_v_PT(643,21.8,"q")
0.002043919161
>>> _Backward3x_v_PT(644,21.1,"r")
0.005251009921
>>> _Backward3x_v_PT(648,21.8,"r")
0.005256844741
>>> _Backward3x_v_PT(635,19.1,"s")
0.001932829079
>>> _Backward3x_v_PT(638,20,"s")
0.001985387227
>>> _Backward3x_v_PT(626,17,"t")
0.008483262001
>>> _Backward3x_v_PT(640,20,"t")
0.006227528101
>>> _Backward3x_v_PT(644.6,21.5,"u")
0.002268366647
>>> _Backward3x_v_PT(646.1,22,"u")
0.002296350553
>>> _Backward3x_v_PT(648.6,22.5,"v")
0.002832373260
>>> _Backward3x_v_PT(647.9,22.3,"v")
0.002811424405
>>> _Backward3x_v_PT(647.5,22.15,"w")
0.003694032281
>>> _Backward3x_v_PT(648.1,22.3,"w")
0.003622226305
>>> _Backward3x_v_PT(648,22.11,"x")
0.004528072649
>>> _Backward3x_v_PT(649,22.3,"x")
0.004556905799
>>> _Backward3x_v_PT(646.84,22,"y")
0.002698354719
>>> _Backward3x_v_PT(647.05,22.064,"y")
0.002717655648
>>> _Backward3x_v_PT(646.89,22,"z")
0.003798732962
>>> _Backward3x_v_PT(647.15,22.064,"z")
0.003701940009
```

iapws.iapws97.**_Region4**(*P*, *x*)

Basic equation for region 4

Parameters

- **P** (*float*) – Pressure, [MPa]
- **x** (*float*) – Vapor quality, [-]

Returns

prop –

Dict with calculated properties. The available properties are:

- T: Saturated temperature, [K]
- P: Saturated pressure, [MPa]
- x: Vapor quality, [-]
- v: Specific volume, [m^3/kg]
- h: Specific enthalpy, [kJ/kg]
- s: Specific entropy, [kJ/kgK]

Return type dict

iapws.iapws97.**_Backward4_T_hs**(*h*, *s*)

Parameters

- **h** (*float*) – Specific enthalpy, [kJ/kg]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **T** – Temperature, [K]

Return type float

References

IAPWS, Revised Supplementary Release on Backward Equations p(h,s) for Region 3, Equations as a Function of h and s for the Region Boundaries, and an Equation Tsat(h,s) for Region 4 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, http://www.iapws.org/relguide/Supp-phs3-2014.pdf. Eq 9

Examples

```
>>> _Backward4_T_hs(1800,5.3)
346.8475498
>>> _Backward4_T_hs(2400,6.0)
425.1373305
>>> _Backward4_T_hs(2500,5.5)
522.5579013
```

iapws.iapws97.**_Region5**(*T*, *P*)

Basic equation for region 5

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]

Returns

prop –

Dict with calculated properties. The available properties are:

- v: Specific volume, [m^3/kg]

- h: Specific enthalpy, [kJ/kg]
- s: Specific entropy, [kJ/kgK]
- cp: Specific isobaric heat capacity, [kJ/kgK]
- cv: Specific isocoric heat capacity, [kJ/kgK]
- w: Speed of sound, [m/s]
- alfav: Cubic expansion coefficient, [1/K]
- kt: Isothermal compressibility, [1/MPa]

Return type dict

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 32-34

Examples

```
>>> _Region5(1500,0.5)["v"]
1.38455090
>>> _Region5(1500,0.5)["h"]
5219.76855
>>> _Region5(1500,0.5)["h"]-500*_Region5(1500,0.5)["v"]
4527.49310
>>> _Region5(1500,30)["s"]
7.72970133
>>> _Region5(1500,30)["cp"]
2.72724317
>>> _Region5(1500,30)["cv"]
2.19274829
>>> _Region5(2000,30)["w"]
1067.36948
>>> _Region5(2000,30)["alfav"]
0.000508830641
>>> _Region5(2000,30)["kt"]
0.0329193892
```

iapws.iapws97.**Region5_cp0**(*Tr*, *Pr*)

Ideal properties for Region 5

Parameters

- **Tr** (*float*) – Reduced temperature, [-]
- **Pr** (*float*) – Reduced pressure, [-]

Returns

prop –

Array with ideal Gibbs energy partial derivatives:

- g: Ideal Specific Gibbs energy, [kJ/kg]
- gp: [g/P]T
- gpp: [^{2}g/P^2]T

- gt: [g/T]P
- gtt: [^{2}g/T^2]P
- gpt: [^{2}g/TP]

Return type array

References

IAPWS, Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam August 2007, http://www.iapws.org/relguide/IF97-Rev.html, Eq 33

iapws.iapws97.**_Bound_TP**(*T*, *P*)
Region definition for input T and P

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]

Returns **region** – IAPWS-97 region code

Return type float

References

Wagner, W; Kretzschmar, H-J: International Steam Tables: Properties of Water and Steam Based on the Industrial Formulation IAPWS-IF97; Springer, 2008; doi: 10.1007/978-3-540-74234-0. Fig. 2.3

iapws.iapws97.**_Bound_Ph**(*P*, *h*)
Region definition for input P y h

Parameters

- **P** (*float*) – Pressure, [MPa]
- **h** (*float*) – Specific enthalpy, [kJ/kg]

Returns **region** – IAPWS-97 region code

Return type float

References

Wagner, W; Kretzschmar, H-J: International Steam Tables: Properties of Water and Steam Based on the Industrial Formulation IAPWS-IF97; Springer, 2008; doi: 10.1007/978-3-540-74234-0. Fig. 2.5

iapws.iapws97.**_Bound_Ps**(*P*, *s*)
Region definition for input P and s

Parameters

- **P** (*float*) – Pressure, [MPa]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **region** – IAPWS-97 region code

Return type float

References

Wagner, W; Kretzschmar, H-J: International Steam Tables: Properties of Water and Steam Based on the Industrial Formulation IAPWS-IF97; Springer, 2008; doi: 10.1007/978-3-540-74234-0. Fig. 2.9

iapws.iapws97.**_Bound_hs**(*h*, *s*)
Region definition for input h and s

Parameters

- **h** (*float*) – Specific enthalpy, [kJ/kg]
- **s** (*float*) – Specific entropy, [kJ/kgK]

Returns **region** – IAPWS-97 region code

Return type float

References

Wagner, W; Kretzschmar, H-J: International Steam Tables: Properties of Water and Steam Based on the Industrial Formulation IAPWS-IF97; Springer, 2008; doi: 10.1007/978-3-540-74234-0. Fig. 2.14

iapws.iapws97.**prop0**(*T*, *P*)
Ideal gas properties

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]

Returns

prop –

Dict with calculated properties. The available properties are:

- v: Specific volume, [m^3/kg]
- h: Specific enthalpy, [kJ/kg]
- s: Specific entropy, [kJ/kgK]
- cp: Specific isobaric heat capacity, [kJ/kgK]
- cv: Specific isocoric heat capacity, [kJ/kgK]
- w: Speed of sound, [m/s]
- alfav: Cubic expansion coefficient, [1/K]
- kt: Isothermal compressibility, [1/MPa]

Return type dict

class iapws.iapws97.**IAPWS97**(***kwargs*)
Class to model a state of liquid water or steam with the IAPWS-IF97

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]
- **h** (*float*) – Specific enthalpy, [kJ/kg]

- **s** (*float*) – Specific entropy, [kJ/kgK]
- **x** (*float*) – Vapor quality, [-]
- **l** (*float, optional*) – Wavelength of light, for refractive index, [μm]

Notes

Definitions options:

- T, P: Not valid for two-phases region
- P, h
- P, s
- h, s
- T, x: Only for two-phases region
- P, x: Only for two-phases region

Returns

prop –

The calculated instance has the following properties:

- P: Pressure, [MPa]
- T: Temperature, [K]
- g: Specific Gibbs free energy, [kJ/kg]
- a: Specific Helmholtz free energy, [kJ/kg]
- v: Specific volume, [m^3/kg]
- rho: Density, [kg/m^3]
- h: Specific enthalpy, [kJ/kg]
- u: Specific internal energy, [kJ/kg]
- s: Specific entropy, [kJ/kg·K]
- cp: Specific isobaric heat capacity, [kJ/kg·K]
- cv: Specific isochoric heat capacity, [kJ/kg·K]
- Z: Compression factor, [-]
- fi: Fugacity coefficient, [-]
- f: Fugacity, [MPa]
- gamma: Isoentropic exponent, [-]
- alfav: Isobaric cubic expansion coefficient, [1/K]
- xkappa: Isothermal compressibility, [1/MPa]
- kappas: Adiabatic compresibility, [1/MPa]
- alfap: Relative pressure coefficient, [1/K]
- betap: Isothermal stress coefficient, [kg/m^3]
- joule: Joule-Thomson coefficient, [K/MPa]

- deltat: Isothermal throttling coefficient, [kJ/kg·MPa]
- region: Region
- v0: Ideal specific volume, [m^3/kg]
- u0: Ideal specific internal energy, [kJ/kg]
- h0: Ideal specific enthalpy, [kJ/kg]
- s0: Ideal specific entropy, [kJ/kg·K]
- a0: Ideal specific Helmholtz free energy, [kJ/kg]
- g0: Ideal specific Gibbs free energy, [kJ/kg]
- cp0: Ideal specific isobaric heat capacity, [kJ/kg·K]
- cv0: Ideal specific isochoric heat capacity [kJ/kg·K]
- w0: Ideal speed of sound, [m/s]
- gamma0: Ideal isoentropic exponent, [-]
- w: Speed of sound, [m/s]
- mu: Dynamic viscosity, [Pa·s]
- nu: Kinematic viscosity, [m^2/s]
- k: Thermal conductivity, [W/m·K]
- alfa: Thermal diffusivity, [m^2/s]
- sigma: Surface tension, [N/m]
- epsilon: Dielectric constant, [-]
- n: Refractive index, [-]
- Prandt: Prandtl number, [-]
- Pr: Reduced Pressure, [-]
- Tr: Reduced Temperature, [-]
- Hvap: Vaporization heat, [kJ/kg]
- Svap: Vaporization entropy, [kJ/kg·K]

Return type dict

Examples

```
>>> water=IAPWS97(T=170+273.15, x=0.5)
>>> water.Liquid.cp, water.Vapor.cp, water.Liquid.w, water.Vapor.w
4.3695 2.5985 1418.3 498.78
```

```
>>> water=IAPWS97(T=325+273.15, x=0.5)
>>> water.P, water.Liquid.v, water.Vapor.v, water.Liquid.h, water.Vapor.h
12.0505 0.00152830 0.0141887 1493.37 2684.48
```

```
>>> water=IAPWS97(T=50+273.15, P=0.0006112127)
>>> water.cp0, water.cv0, water.h0, water.s0, water.w0
1.8714 1.4098 2594.66 9.471 444.93
```

Attributes

calculable	Check if class is calculable by its kwargs

Methods

__call__(**kwargs)	Invoke the solver.
calculo()	Calculate procedure
derivative(z, x, y, fase)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, u, h, s, g, a
fill(fase, estado)	Fill phase properties

status = 0

msg = 'Unknown variables'

kwargs = {'P': 0.0, 'T': 0.0, 'h': None, 'l': 0.5893, 's': None, 'v': 0.0, 'x': N

calculable
Check if class is calculable by its kwargs

calculo()
Calculate procedure

fill(*fase*, *estado*)
Fill phase properties

derivative(*z*, *x*, *y*, *fase*)
Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, u, h, s, g, a

class iapws.iapws97.**IAPWS97_PT**(*P*, *T*)
Derivated class for direct P and T input

Attributes

calculable	Check if class is calculable by its kwargs

Methods

__call__(**kwargs)	Invoke the solver.
calculo()	Calculate procedure
derivative(z, x, y, fase)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, u, h, s, g, a
fill(fase, estado)	Fill phase properties

class iapws.iapws97.**IAPWS97_Ph**(*P*, *h*)
Derivated class for direct P and h input

Attributes

calculable	Check if class is calculable by its kwargs

Methods

`__call__`(**kwargs)	Invoke the solver.
`calculo`()	Calculate procedure
`derivative`(z, x, y, fase)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, u, h, s, g, a
`fill`(fase, estado)	Fill phase properties

class iapws.iapws97.**IAPWS97_Ps**(*P*, *s*)

Derivated class for direct P and s input

Attributes

calculable Check if class is calculable by its kwargs

Methods

`__call__`(**kwargs)	Invoke the solver.
`calculo`()	Calculate procedure
`derivative`(z, x, y, fase)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, u, h, s, g, a
`fill`(fase, estado)	Fill phase properties

class iapws.iapws97.**IAPWS97_Px**(*P*, *x*)

Derivated class for direct P and x input

Attributes

calculable Check if class is calculable by its kwargs

Methods

`__call__`(**kwargs)	Invoke the solver.
`calculo`()	Calculate procedure
`derivative`(z, x, y, fase)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, u, h, s, g, a
`fill`(fase, estado)	Fill phase properties

class iapws.iapws97.**IAPWS97_Tx**(*T*, *x*)

Derivated class for direct T and x input

Attributes

calculable Check if class is calculable by its kwargs

Methods

`__call__`(**kwargs)	Invoke the solver.
`calculo`()	Calculate procedure
`derivative`(z, x, y, fase)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, u, h, s, g, a
`fill`(fase, estado)	Fill phase properties

5.1.5 iapws.iapws08 module

IAPWS standard for Seawater IAPWS08 and related functionality. The module include:

SeaWater: Global module class with all the functionality integrated

Other functionality:

- *_Tb()*: Boiling temperature of seawater
- *_Tf()*: Freezing temperature of seawater
- *_Triple()*: Triple point properties of seawater
- *_OsmoticPressure()*: Osmotic pressure of seawater
- *_ThCond_SeaWater()*: Thermal conductivity of seawater
- *_Tension_SeaWater()*: Surface tension of seawater
- *_solNa2SO4()*: Solubility of sodium sulfate in aqueous mixtures of sodium chloride and sulfuric acid
- *_critNaCl()*: Critical locus of aqueous solutions of sodium chloride

class iapws.iapws08.**SeaWater**(***kwargs*)
Class to model seawater with standard IAPWS-08

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]
- **S** (*float*) – Salinity, [kg/kg]
- **fast** (*bool, default False*) – Use the Supplementary release SR7-09 to speed up the calculation
- **IF97** (*bool, default False*) – Use the Advisory Note No. 5 with industrial formulation

Returns

- **rho** (*float*) – Density, [kg/m^3]
- **v** (*float*) – Specific volume, [m^3/kg]
- **h** (*float*) – Specific enthalpy, [kJ/kg]
- **s** (*float*) – Specific entropy, [kJ/kg·K]
- **u** (*float*) – Specific internal energy, [kJ/kg]
- **g** (*float*) – Specific Gibbs free energy, [kJ/kg]
- **a** (*float*) – Specific Helmholtz free energy, [kJ/kg]
- **cp** (*float*) – Specific isobaric heat capacity, [kJ/kg·K]
- **cv** (*float*) – Specific isochoric heat capacity, [kJ/kg·K]
- **gt** (*float*) – Derivative Gibbs energy with temperature, [kJ/kg·K]
- **gp** (*float*) – Derivative Gibbs energy with pressure, [m^3/kg]
- **gtt** (*float*) – Derivative Gibbs energy with temperature square, [kJ/kg·K^2]
- **gtp** (*float*) – Derivative Gibbs energy with pressure and temperature, [m^3/kg·K]
- **gpp** (*float*) – Derivative Gibbs energy with temperature square, [m^3/kg·MPa]

- **gs** (*float*) – Derivative Gibbs energy with salinity, [kJ/kg]
- **gsp** (*float*) – Derivative Gibbs energy with salinity and pressure, [m^3/kg]
- **alfav** (*float*) – Thermal expansion coefficient, [1/K]
- **betas** (*float*) – Isentropic temperature-pressure coefficient, [K/MPa]
- **xkappa** (*float*) – Isothermal compressibility, [1/MPa]
- **ks** (*float*) – Isentropic compressibility, [1/MPa]
- **w** (*float*) – Sound Speed, [m/s]
- **k** (*float*) – Thermal conductivity, [W/m·K]
- **sigma** (*float*) – Surface tension, [N/m]
- **m** (*float*) – Molality of seawater, [mol/kg]
- **mu** (*float*) – Relative chemical potential, [kJ/kg]
- **muw** (*float*) – Chemical potential of H2O, [kJ/kg]
- **mus** (*float*) – Chemical potential of sea salt, [kJ/kg]
- **osm** (*float*) – Osmotic coefficient, [-]
- **haline** (*float*) – Haline contraction coefficient, [kg/kg]

Notes

`Warning` if input isn't in limit:

- 261 T 353
- 0 < P 100
- 0 S 0.12

References

IAPWS, Release on the IAPWS Formulation 2008 for the Thermodynamic Properties of Seawater, http://www.iapws.org/relguide/Seawater.html

IAPWS, Supplementary Release on a Computationally Efficient Thermodynamic Formulation for Liquid Water for Oceanographic Use, http://www.iapws.org/relguide/OceanLiquid.html

IAPWS, Guideline on the Thermal Conductivity of Seawater, http://www.iapws.org/relguide/Seawater-ThCond.html

IAPWS, Guideline on the Surface Tension of Seawater, http://www.iapws.org/relguide/Seawater-Surf.html

IAPWS, Revised Advisory Note No. 3: Thermodynamic Derivatives from IAPWS Formulations, http://www.iapws.org/relguide/Advise3.pdf

IAPWS, Advisory Note No. 5: Industrial Calculation of the Thermodynamic Properties of Seawater, http://www.iapws.org/relguide/Advise5.html

Examples

```
>>> salt = iapws.SeaWater(T=300, P=1, S=0.04)
>>> salt.rho
1026.7785717245113
>>> salt.gs
88.56221805501536
>>> salt.haline
0.7311487666026304
```

Methods

__call__(**kwargs)	Make instance callable to can add input parameter one to one
calculo()	Calculate procedure
derivative(z, x, y)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, u, h, s, g, a

status = 0

msg = 'Undefined'

kwargs = {'IF97': False, 'P': 0.0, 'S': None, 'T': 0.0, 'fast': False}

calculo()
Calculate procedure

derivative(*z*, *x*, *y*)
Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, u, h, s, g, a

classmethod _water(*T*, *P*)
Get properties of pure water, Table4 pag 8

classmethod _waterIF97(*T*, *P*)

classmethod _waterSupp(*T*, *P*)
Get properties of pure water using the supplementary release SR7-09, Table4 pag 6

classmethod _saline(*T*, *P*, *S*)
Eq 4

iapws.iapws08.**_Tb**(*P*, *S*)
Procedure to calculate the boiling temperature of seawater

Parameters

- **P** (*float*) – Pressure, [MPa]
- **S** (*float*) – Salinity, [kg/kg]

Returns **Tb** – Boiling temperature, [K]

Return type float

References

IAPWS, Advisory Note No. 5: Industrial Calculation of the Thermodynamic Properties of Seawater, http://www.iapws.org/relguide/Advise5.html, Eq 7

iapws.iapws08.**_Tf**(*P*, *S*)
Procedure to calculate the freezing temperature of seawater

Parameters

- **P** (*float*) – Pressure, [MPa]
- **S** (*float*) – Salinity, [kg/kg]

Returns **Tf** – Freezing temperature, [K]

Return type float

References

IAPWS, Advisory Note No. 5: Industrial Calculation of the Thermodynamic Properties of Seawater, http://www.iapws.org/relguide/Advise5.html, Eq 12

iapws.iapws08.**_Triple**(*S*)
Procedure to calculate the triple point pressure and temperature for seawater

Parameters **S** (*float*) – Salinity, [kg/kg]

Returns

prop –

Dictionary with the triple point properties:

- Tt: Triple point temperature, [K]
- Pt: Triple point pressure, [MPa]

Return type dict

References

IAPWS, Advisory Note No. 5: Industrial Calculation of the Thermodynamic Properties of Seawater, http://www.iapws.org/relguide/Advise5.html, Eq 7

iapws.iapws08.**_OsmoticPressure**(*T*, *P*, *S*)
Procedure to calculate the osmotic pressure of seawater

Parameters

- **T** (*float*) – Tmperature, [K]
- **P** (*float*) – Pressure, [MPa]
- **S** (*float*) – Salinity, [kg/kg]

Returns **Posm** – Osmotic pressure, [MPa]

Return type float

References

IAPWS, Advisory Note No. 5: Industrial Calculation of the Thermodynamic Properties of Seawater, http://www.iapws.org/relguide/Advise5.html, Eq 15

iapws.iapws08.**_ThCond_SeaWater**(*T*, *P*, *S*)
Equation for the thermal conductivity of seawater

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]
- **S** (*float*) – Salinity, [kg/kg]

Returns **k** – Thermal conductivity excess relative to that of the pure water, [W/mK]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- 273.15 T 523.15
- 0 P 140
- 0 S 0.17

Examples

```
>>> _ThCond_Seawater(293.15, 0.1, 0.035)
-0.00418604
```

References

IAPWS, Guideline on the Thermal Conductivity of Seawater, http://www.iapws.org/relguide/Seawater-ThCond.html

iapws.iapws08.**_Tension_SeaWater**(*T*, *S*)

Equation for the surface tension of seawater

Parameters

- **T** (*float*) – Temperature, [K]
- **S** (*float*) – Salinity, [kg/kg]

Returns σ – Surface tension, [N/m]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- 0 S 0.131 for 274.15 T 365.15
- 0 S 0.038 for 248.15 T 274.15

Examples

```
>>> _Tension_Seawater(253.15, 0.035)
-0.07922517961
```

References

IAPWS, Guideline on the Surface Tension of Seawater, http://www.iapws.org/relguide/Seawater-Surf.html

iapws.iapws08.**_solNa2SO4**(*T*, *mH2SO4*, *mNaCl*)

Equation for the solubility of sodium sulfate in aqueous mixtures of sodium chloride and sulfuric acid

Parameters

- **T** (*float*) – Temperature, [K]
- **mH2SO4** (*float*) – Molality of sufuric acid, [mol/kg(water)]
- **mNaCl** (*float*) – Molality of sodium chloride, [mol/kg(water)]

Returns **S** – Molal solutility of sodium sulfate, [mol/kg(water)]

Return type float

Notes

Raise NotImplementedError if input isn't in limit:

- 523.15 T 623.15
- 0 mH2SO4 0.75
- 0 mNaCl 2.25

Examples

```
>>> _solNa2SO4(523.15, 0.25, 0.75)
2.68
```

References

IAPWS, Solubility of Sodium Sulfate in Aqueous Mixtures of Sodium Chloride and Sulfuric Acid from Water to Concentrated Solutions, http://www.iapws.org/relguide/na2so4.pdf

iapws.iapws08.**_critNaCl**(*x*)

Equation for the critical locus of aqueous solutions of sodium chloride

Parameters **x** (*float*) – Mole fraction of NaCl, [-]

Returns

prop –

A dictionary withe the properties:

- Tc: critical temperature, [K]
- Pc: critical pressure, [MPa]
- rhoc: critical density, [kg/m^3]

Return type dict

Notes

Raise `NotImplementedError` if input isn't in limit:

- 0 x 0.12

Examples

```
>>> _critNaCl(0.1)
975.571016
```

References

IAPWS, Revised Guideline on the Critical Locus of Aqueous Solutions of Sodium Chloride, http://www.iapws.org/relguide/critnacl.html

5.1.6 iapws.ammonia module

Module with Ammonia-water mixture properties and related properties. The module include:

- *NH3*: Multiparameter equation of state for ammonia
- *H2ONH3*: Thermodynamic properties of ammonia-water mixtures
- *Ttr*: Triple point of ammonia-water mixtures

class iapws.ammonia.**NH3**(***kwargs*)

Multiparameter equation of state for ammonia for internal procedures, see MEoS base class

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]
- **rho** (*float*) – Density, [kg/m^3]
- **v** (*float*) – Specific volume, [m^3/kg]
- **h** (*float*) – Specific enthalpy, [kJ/kg]
- **s** (*float*) – Specific entropy, [kJ/kgK]
- **u** (*float*) – Specific internal energy, [kJ/kg]
- **x** (*float*) – Vapor quality, [-]
- **l** (*float, optional*) – Wavelength of light, for refractive index, [μm]
- **rho0** (*float, optional*) – Initial value of density, to improve iteration, [kg/m^3]
- **T0** (*float, optional*) – Initial value of temperature, to improve iteration, [K]
- **x0** (*Initial value of vapor quality, necessary in bad input pair definition*) – where there are two valid solution (T-h, T-s)

Notes

- It needs two incoming properties of T, P, rho, h, s, u.
- v as a alternate input parameter to rho
- T-x, P-x, preferred input pair to specified a point in two phases region

The calculated instance has the following properties:

- P: Pressure, [MPa]
- T: Temperature, [K]
- x: Vapor quality, [-]
- g: Specific Gibbs free energy, [kJ/kg]
- a: Specific Helmholtz free energy, [kJ/kg]
- v: Specific volume, [m^3/kg]
- r: Density, [kg/m^3]
- h: Specific enthalpy, [kJ/kg]
- u: Specific internal energy, [kJ/kg]
- s: Specific entropy, [kJ/kg·K]
- cp: Specific isobaric heat capacity, [kJ/kg·K]
- cv: Specific isochoric heat capacity, [kJ/kg·K]
- cp_cv: Heat capacity ratio, [-]
- Z: Compression factor, [-]
- fi: Fugacity coefficient, [-]
- f: Fugacity, [MPa]
- gamma: Isoentropic exponent, [-]
- alfav: Isobaric cubic expansion coefficient, [1/K]
- kappa: Isothermal compressibility, [1/MPa]
- kappas: Adiabatic compresibility, [1/MPa]
- alfap: Relative pressure coefficient, [1/K]
- betap: Isothermal stress coefficient, [kg/m^3]
- joule: Joule-Thomson coefficient, [K/MPa]
- betas: Isoentropic temperature-pressure coefficient, [-]
- Gruneisen: Gruneisen parameter, [-]
- virialB: Second virial coefficient, [m^3/kg]
- virialC: Third virial coefficient, [m^6/kg^2]
- dpdT_rho: Derivatives, dp/dT at constant rho, [MPa/K]
- dpdrho_T: Derivatives, dp/drho at constant T, [MPa·m^3/kg]
- drhodT_P: Derivatives, drho/dT at constant P, [kg/m^3·K]

- drhodP_T: Derivatives, drho/dP at constant T, [kg/m^3·MPa]
- dhdT_rho: Derivatives, dh/dT at constant rho, [kJ/kg·K]
- dhdP_T: Isothermal throttling coefficient, [kJ/kg·MPa]
- dhdT_P: Derivatives, dh/dT at constant P, [kJ/kg·K]
- dhdrho_T: Derivatives, dh/drho at constant T, [kJ·m^3/kg^2]
- dhdrho_P: Derivatives, dh/drho at constant P, [kJ·m^3/kg^2]
- dhdP_rho: Derivatives, dh/dP at constant rho, [kJ/kg·MPa]
- kt: Isothermal Expansion Coefficient, [-]
- ks: Adiabatic Compressibility, [1/MPa]
- Ks: Adiabatic bulk modulus, [MPa]
- Kt: Isothermal bulk modulus, [MPa]
- v0: Ideal specific volume, [m^3/kg]
- rho0: Ideal gas density, [kg/m^3]
- u0: Ideal specific internal energy, [kJ/kg]
- h0: Ideal specific enthalpy, [kJ/kg]
- s0: Ideal specific entropy, [kJ/kg·K]
- a0: Ideal specific Helmholtz free energy, [kJ/kg]
- g0: Ideal specific Gibbs free energy, [kJ/kg]
- cp0: Ideal specific isobaric heat capacity, [kJ/kg·K]
- cv0: Ideal specific isochoric heat capacity, [kJ/kg·K]
- w0: Ideal speed of sound, [m/s]
- gamma0: Ideal isoentropic exponent, [-]
- w: Speed of sound, [m/s]
- mu: Dynamic viscosity, [Pa·s]
- nu: Kinematic viscosity, [m^2/s]
- k: Thermal conductivity, [W/m·K]
- alfa: Thermal diffusivity, [m^2/s]
- sigma: Surface tension, [N/m]
- epsilon: Dielectric constant, [-]
- n: Refractive index, [-]
- Prandt: Prandtl number, [-]
- Pr: Reduced Pressure, [-]
- Tr: Reduced Temperature, [-]
- Hvap: Vaporization heat, [kJ/kg]
- Svap: Vaporization entropy, [kJ/kg·K]
- Z_rho: $(Z-1)/\rho$, [m^3/kg]

- IntP: Internal pressure, [MPa]
- invT: Negative reciprocal temperature, [1/K]
- hInput: Specific heat input, [kJ/kg]

References

Baehr, H.D., Tillner-Roth, R.; Thermodynamic Properties of Environmentally Acceptable Refrigerants: Equations of State and Tables for Ammonia, R22, R134a, R152a, and R123. Springer-Verlag, Berlin, 1994. http://doi.org/10.1007/978-3-642-79400-1

Attributes

CP

Gruneisen

IntP

Ks

Kt

Prandt

Z

Z_rho

a

alfa

alfap

alfav

betap

betas

`calculable` Check if inputs are enough to define state

cp

cp_cv

cv

dhdP_T

dhdP_rho

dhdT_P

dhdT_rho

dhdrho_P

dhdrho_T

dpdT_rho

dpdrho_T

drhodP_T

drhodT_P

epsilon

f

fi

g

gamma

h

hInput

joule

k

kappa

ks

kt

mu

n

nu

rho

s

u

v

w

Methods

`__call__`(**kwargs)	Make instance callable to can add input parameter one to one
`calculo`()	Calculate procedure
`derivative`(z, x, y, fase)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, rho, u, h, s, g, a
`fill`(fase, estado)	Fill phase properties

```
name = 'ammonia'
CASNumber = '7664-41-7'
formula = 'NH3'
synonym = 'R-717'
rhoc = 225.0
Tc = 405.4
Pc = 11.333
M = 17.03026
Tt = 195.495
```

```
Tb = 239.823
f_acent = 0.25601
momentoDipolar = 1.47
Fi0 = {'ao_exp': [], 'ao_hyp': [], 'ao_log': [1, -1], 'ao_pow': [-15.81502, 4.2557
_constants = {'R': 8.314471, 'c2': [1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3],
_melting = {'Pref': 1000, 'Tmax': 700.0, 'Tmin': 195.495, 'Tref': 195.495, 'a1':
_surf = {'exp': [1.211, 5.585], 'sigma': [0.1028, -0.09453]}
_Pv = {'ao': [-7.0993, -2.433, 8.7591, -6.4091, -2.1185], 'eq': 5, 'exp': [1.0, 1.5
_rhoL = {'ao': [34.488, -128.49, 173.82, -106.99, 30.339], 'eq': 1, 'exp': [0.58, 0
_rhoG = {'ao': [-0.38435, -4.0846, -6.6634, -31.881, 213.06, -246.48], 'eq': 3, 'exp
```

_visco(*rho*, *T*, *fase=None*)
Equation for the Viscosity

Parameters

- **rho** (*float*) – Density [kg/m^3]
- **T** (*float*) – Temperature [K]

Returns **mu** – Viscosity [Pa·s]

Return type float

References

Fenghour, A., Wakeham, W.A., Vesovic, V., Watson, J.T.R., Millat, J., and Vogel, E., The viscosity of ammonia, J. Phys. Chem. Ref. Data 24, 1649 (1995). doi:10.1063/1.555961

_thermo(*rho*, *T*, *fase*)
Equation for the thermal conductivity

Parameters

- **rho** (*float*) – Density, [kg/m^3]
- **T** (*float*) – Temperature, [K]
- **fase** (*dict*) – phase properties

Returns **k** – Thermal conductivity [W/mK]

Return type float

References

Tufeu, R., Ivanov, D.Y., Garrabos, Y., and Le Neindre, B., Thermal conductivity of ammonia in a large temperature and pressure range including the critical region, Ber. Bunsenges. Phys. Chem., 88:422-427, 1984. doi:10.1002/bbpc.19840880421

class iapws.ammonia.**H2ONH3**
Ammonia-water mixtures.

_prop(*rho*, *T*, *x*)
Thermodynamic properties of ammonia-water mixtures

Parameters

- **T** (*float*) – Temperature [K]
- **rho** (*float*) – Density [kg/m^3]
- **x** (*float*) – Mole fraction of ammonia in mixture [mol/mol]

Returns

prop –

Dictionary with thermodynamic properties of ammonia-water mixtures:

- M: Mixture molecular mass, [g/mol]
- P: Pressure, [MPa]
- u: Specific internal energy, [kJ/kg]
- s: Specific entropy, [kJ/kgK]
- h: Specific enthalpy, [kJ/kg]
- a: Specific Helmholtz energy, [kJ/kg]
- g: Specific gibbs energy, [kJ/kg]
- cv: Specific isochoric heat capacity, [kJ/kgK]
- cp: Specific isobaric heat capacity, [kJ/kgK]
- w: Speed of sound, [m/s]
- fugH2O: Fugacity of water, [-]
- fugNH3: Fugacity of ammonia, [-]

Return type dict

References

IAPWS, Guideline on the IAPWS Formulation 2001 for the Thermodynamic Properties of Ammonia-Water Mixtures, http://www.iapws.org/relguide/nh3h2o.pdf, Table 4

_phi0(*rho*, *T*, *x*)

Ideal gas Helmholtz energy of binary mixtures and derivatives

Parameters

- **rho** (*float*) – Density, [kg/m^3]
- **T** (*float*) – Temperature, [K]
- **x** (*float*) – Mole fraction of ammonia in mixture, [mol/mol]

Returns

prop – Dictionary with ideal adimensional helmholtz energy and derivatives:

- tau: the adimensional temperature variable, [-]
- delta: the adimensional density variable, [-]
- fio,[-]
- fiot: [fio/τ]δ [-]
- fiod: [fio/δ]τ [-]

- fiott: $[^2\text{fio}/\tau^2]\delta$ [-]
- fiodt: $[^2\text{fio}/\tau\delta]$ [-]
- fiodd: $[^2\text{fio}/\delta^2]\tau$ [-]

Return type dict

References

IAPWS, Guideline on the IAPWS Formulation 2001 for the Thermodynamic Properties of Ammonia-Water Mixtures, http://www.iapws.org/relguide/nh3h2o.pdf, Eq 2

_phir (*rho*, *T*, *x*)
Residual contribution to the free Helmholtz energy

Parameters

- **rho** (*float*) – Density, [kg/m^3]
- **T** (*float*) – Temperature, [K]
- **x** (*float*) – Mole fraction of ammonia in mixture, [mol/mol]

Returns

prop – dictionary with residual adimensional helmholtz energy and derivatives:

- tau: the adimensional temperature variable, [-]
- delta: the adimensional density variable, [-]
- fir, [-]
- firt: $[\text{fir}/\tau]\delta$,x [-]
- fird: $[\text{fir}/\delta]\tau$,x [-]
- firtt: $[^2\text{fir}/\tau^2]\delta$,x [-]
- firdt: $[^2\text{fir}/\tau\delta]$x [-]
- firdd: $[^2\text{fir}/\delta^2]\tau$,x [-]
- firx: $[\text{fir}/\text{x}]\tau,\delta$ [-]
- F: Function for fugacity calculation, [-]

Return type dict

References

IAPWS, Guideline on the IAPWS Formulation 2001 for the Thermodynamic Properties of Ammonia-Water Mixtures, http://www.iapws.org/relguide/nh3h2o.pdf, Eq 3

_Dphir (*tau*, *delta*, *x*)
Departure function to the residual contribution to the free Helmholtz energy

Parameters

- **tau** (*float*) – Adimensional temperature, [-]
- **delta** (*float*) – Adimensional density, [-]
- **x** (*float*) – Mole fraction of ammonia in mixture, [mol/mol]

Returns

prop – Dictionary with departure contribution to the residual adimensional helmholtz energy and derivatives:

- fir [-]
- firt: $[\Delta \text{fir}/\tau]\delta$,x [-]
- fird: $[\Delta \text{fir}/\delta]\tau$,x [-]
- firtt: $[^2\Delta \text{fir}/\tau^2]\delta$,x [-]
- firdt: $[^2\Delta \text{fir}/\tau\delta]$x [-]
- firdd: $[^2\Delta \text{fir}/\delta^2]\tau$,x [-]
- firx: $[\Delta \text{fir}/\text{x}]\tau,\delta$ [-]

Return type dict

References

IAPWS, Guideline on the IAPWS Formulation 2001 for the Thermodynamic Properties of Ammonia-Water Mixtures, http://www.iapws.org/relguide/nh3h2o.pdf, Eq 8

`iapws.ammonia.`**`Ttr`**`(x)`

Equation for the triple point of ammonia-water mixture

Parameters **x** (*float*) – Mole fraction of ammonia in mixture, [mol/mol]

Returns **Ttr** – Triple point temperature, [K]

Return type float

Notes

Raise `NotImplementedError` if input isn't in limit:

- 0 x 1

References

IAPWS, Guideline on the IAPWS Formulation 2001 for the Thermodynamic Properties of Ammonia-Water Mixtures, http://www.iapws.org/relguide/nh3h2o.pdf, Eq 9

5.1.7 iapws.humidAir module

Module with Air-water mixture properties and related properties. The module include:

- *_virial()*: Virial equations for humid air
- *_fugacity()*: Fugacity equation for humid air
- *MEoSBlend*: Special MEoS subclass to implement pseudocomponent blend with ancillary dew and bubble point
- *Air*: Multiparameter equation of state for Air as pseudocomponent
- *HumidAir*: Humid air mixture with complete functionality

iapws.humidAir.**_virial**(*T*)

Virial equations for humid air

Parameters **T** (*float*) – Temperature [K]

Returns

prop –

Dictionary with critical coefficient:

- Baa: Second virial coefficient of dry air, [m^3/mol]
- Baw: Second air-water cross virial coefficient, [m^3/mol]
- Bww: Second virial coefficient of water, [m^3/mol]
- Caaa: Third virial coefficient of dry air, [m^6/mol]
- Caaw: Third air-water cross virial coefficient, [m^6/mol]
- Caww: Third air-water cross virial coefficient, [m^6/mol]
- Cwww: Third virial coefficient of dry air, [m^6/mol]
- Bawt: dBaw/dT, [m^3/molK]
- Bawtt: d^2Baw/dT^2, [m^3/molK^2]
- Caawt: dCaaw/dT, [m^6/molK]
- Caawtt: d^2Caaw/dT^2, [m^6/molK^2]
- Cawwt: dCaww/dT, [m^6/molK]
- Cawwtt: d^2Caww/dT^2, [m^6/molK^2]

Return type dict

Notes

Raise `Warning` if T isn't in range of validity:

- Baa: 60 T 2000
- Baw: 130 T 2000
- Bww: 130 T 1273
- Caaa: 60 T 2000
- Caaw: 193 T 493
- Caww: 173 T 473
- Cwww: 130 T 1273

Examples

```
>>> _virial(200)["Baa"]
-3.92722567e-5
```

References

IAPWS, Guideline on a Virial Equation for the Fugacity of H2O in Humid Air, http://www.iapws.org/relguide/VirialFugacity.html

IAPWS, Guideline on an Equation of State for Humid Air in Contact with Seawater and Ice, Consistent with the IAPWS Formulation 2008 for the Thermodynamic Properties of Seawater, Table 10, http://www.iapws.org/relguide/SeaAir.html

iapws.humidAir.**_fugacity**(*T*, *P*, *x*)

Fugacity equation for humid air

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]
- **x** (*float*) – Mole fraction of water-vapor, [-]

Returns **fv** – fugacity coefficient, [MPa]

Return type float

Notes

Raise NotImplementedError if input isn't in range of validity:

- 193 T 473
- 0 P 5
- 0 x 1

Really the xmax is the xsaturation but isn't implemented

Examples

```
>>> _fugacity(300, 1, 0.1)
0.0884061686
```

References

IAPWS, Guideline on a Virial Equation for the Fugacity of H2O in Humid Air, http://www.iapws.org/relguide/VirialFugacity.html

class iapws.humidAir.**MEoSBlend**(***kwargs*)

Special meos class to implement pseudocomponent blend and defining its ancillary dew and bubble point

Attributes

CP

Gruneisen

IntP

Ks

Kt

Prandt

Z

Z_rho

a

alfa

alfap

alfav

betap

betas

calculable Check if inputs are enough to define state

cp

cp_cv

cv

dhdP_T

dhdP_rho

dhdT_P

dhdT_rho

dhdrho_P

dhdrho_T

dpdT_rho

dpdrho_T

drhodP_T

drhodT_P

epsilon

f

fi

g

gamma

h

hInput

joule

k

kappa

ks

kt

mu

n

nu

rho

s

u

v

w

Methods

`__call__`(**kwargs)	Make instance callable to can add input parameter one to one
`calculo`()	Calculate procedure
`derivative`(z, x, y, fase)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, rho, u, h, s, g, a
`fill`(fase, estado)	Fill phase properties

classmethod _dewP(*T*)
Using ancillary equation return the pressure of dew point

classmethod _bubbleP(*T*)
Using ancillary equation return the pressure of bubble point

class iapws.humidAir.**Air**(***kwargs*)
Multiparameter equation of state for Air as pseudocomponent for internal procedures, see MEoS base class

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]
- **rho** (*float*) – Density, [kg/m^3]
- **v** (*float*) – Specific volume, [m^3/kg]
- **h** (*float*) – Specific enthalpy, [kJ/kg]
- **s** (*float*) – Specific entropy, [kJ/kgK]
- **u** (*float*) – Specific internal energy, [kJ/kg]
- **x** (*float*) – Vapor quality, [-]
- **l** (*float, optional*) – Wavelength of light, for refractive index, [μm]
- **rho0** (*float, optional*) – Initial value of density, to improve iteration, [kg/m^3]
- **T0** (*float, optional*) – Initial value of temperature, to improve iteration, [K]
- **x0** (*Initial value of vapor quality, necessary in bad input pair definition*) – where there are two valid solution (T-h, T-s)

Notes

- It needs two incoming properties of T, P, rho, h, s, u.

- v as a alternate input parameter to rho
- T-x, P-x, preferred input pair to specified a point in two phases region

The calculated instance has the following properties:

- P: Pressure, [MPa]
- T: Temperature, [K]
- x: Vapor quality, [-]
- g: Specific Gibbs free energy, [kJ/kg]
- a: Specific Helmholtz free energy, [kJ/kg]
- v: Specific volume, [m^3/kg]
- r: Density, [kg/m^3]
- h: Specific enthalpy, [kJ/kg]
- u: Specific internal energy, [kJ/kg]
- s: Specific entropy, [kJ/kg·K]
- cp: Specific isobaric heat capacity, [kJ/kg·K]
- cv: Specific isochoric heat capacity, [kJ/kg·K]
- cp_cv: Heat capacity ratio, [-]
- Z: Compression factor, [-]
- fi: Fugacity coefficient, [-]
- f: Fugacity, [MPa]
- gamma: Isoentropic exponent, [-]
- alfav: Isobaric cubic expansion coefficient, [1/K]
- kappa: Isothermal compressibility, [1/MPa]
- kappas: Adiabatic compresibility, [1/MPa]
- alfap: Relative pressure coefficient, [1/K]
- betap: Isothermal stress coefficient, [kg/m^3]
- joule: Joule-Thomson coefficient, [K/MPa]
- betas: Isoentropic temperature-pressure coefficient, [-]
- Gruneisen: Gruneisen parameter, [-]
- virialB: Second virial coefficient, [m^3/kg]
- virialC: Third virial coefficient, [m^6/kg^2]
- dpdT_rho: Derivatives, dp/dT at constant rho, [MPa/K]
- dpdrho_T: Derivatives, dp/drho at constant T, [MPa·m^3/kg]
- drhodT_P: Derivatives, drho/dT at constant P, [kg/m^3·K]
- drhodP_T: Derivatives, drho/dP at constant T, [kg/m^3·MPa]
- dhdT_rho: Derivatives, dh/dT at constant rho, [kJ/kg·K]
- dhdP_T: Isothermal throttling coefficient, [kJ/kg·MPa]

- dhdT_P: Derivatives, dh/dT at constant P, [kJ/kg·K]
- dhdrho_T: Derivatives, dh/drho at constant T, [kJ·m^3/kg^2]
- dhdrho_P: Derivatives, dh/drho at constant P, [kJ·m^3/kg^2]
- dhdP_rho: Derivatives, dh/dP at constant rho, [kJ/kg·MPa]
- kt: Isothermal Expansion Coefficient, [-]
- ks: Adiabatic Compressibility, [1/MPa]
- Ks: Adiabatic bulk modulus, [MPa]
- Kt: Isothermal bulk modulus, [MPa]
- v0: Ideal specific volume, [m^3/kg]
- rho0: Ideal gas density, [kg/m^3]
- u0: Ideal specific internal energy, [kJ/kg]
- h0: Ideal specific enthalpy, [kJ/kg]
- s0: Ideal specific entropy, [kJ/kg·K]
- a0: Ideal specific Helmholtz free energy, [kJ/kg]
- g0: Ideal specific Gibbs free energy, [kJ/kg]
- cp0: Ideal specific isobaric heat capacity, [kJ/kg·K]
- cv0: Ideal specific isochoric heat capacity, [kJ/kg·K]
- w0: Ideal speed of sound, [m/s]
- gamma0: Ideal isoentropic exponent, [-]
- w: Speed of sound, [m/s]
- mu: Dynamic viscosity, [Pa·s]
- nu: Kinematic viscosity, [m^2/s]
- k: Thermal conductivity, [W/m·K]
- alfa: Thermal diffusivity, [m^2/s]
- sigma: Surface tension, [N/m]
- epsilon: Dielectric constant, [-]
- n: Refractive index, [-]
- Prandt: Prandtl number, [-]
- Pr: Reduced Pressure, [-]
- Tr: Reduced Temperature, [-]
- Hvap: Vaporization heat, [kJ/kg]
- Svap: Vaporization entropy, [kJ/kg·K]
- Z_rho: $(Z-1)/\rho$, [m^3/kg]
- IntP: Internal pressure, [MPa]
- invT: Negative reciprocal temperature, [1/K]
- hInput: Specific heat input, [kJ/kg]

References

Lemmon, E.W., Jacobsen, R.T, Penoncello, S.G., Friend, D.G.; Thermodynamic Properties of Air and Mixtures of Nitrogen, Argon, and Oxygen From 60 to 2000 K at Pressures to 2000 MPa. J. Phys. Chem. Ref. Data 29, 331 (2000). http://dx.doi.org/10.1063/1.1285884

Attributes

CP

Gruneisen

IntP

Ks

Kt

Prandt

Z

Z_rho

a

alfa

alfap

alfav

betap

betas

`calculable` Check if inputs are enough to define state

cp

cp_cv

cv

dhdP_T

dhdP_rho

dhdT_P

dhdT_rho

dhdrho_P

dhdrho_T

dpdT_rho

dpdrho_T

drhodP_T

drhodT_P

epsilon

f

fi

g

gamma

h

hInput

joule

k

kappa

ks

kt

mu

n

nu

rho

s

u

v

w

Methods

`__call__`(**kwargs)	Make instance callable to can add input parameter one to one
`calculo`()	Calculate procedure
`derivative`(z, x, y, fase)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, rho, u, h, s, g, a
`fill`(fase, estado)	Fill phase properties

```
name = 'air'
CASNumber = '1'
formula = 'N2+Ar+O2'
synonym = 'R-729'
rhoc = 302.622436442
Tc = 132.6306
Pc = 3.786
M = 28.96546
Tt = 59.75
Tb = 78.903
f_acent = 0.0335
momentoDipolar = 0.0
```

```
Fi0 = {'ao_exp': [0.791309509, 0.212236768], 'ao_exp2': [-0.197938904], 'ao_log': [
_constants = {'R': 8.31451, 'Tref': 132.6312, 'c2': [1, 1, 1, 1, 2, 2, 2, 3, 3], 'd1
_blend = {'Pj': 3.78502, 'Tj': 132.6312, 'bubble': {'i': [1, 2, 3, 4, 5, 6], 'n':
_melting = {'Pref': 5.265, 'Tmax': 2000.0, 'Tmin': 59.75, 'Tref': 78.903, 'a1': [
_surf = {'exp': [1.28], 'sigma': [0.03046]}
_rhoG = {'ao': [-2.0466, -4.752, -13.259, -47.652], 'eq': 3, 'exp': [0.41, 1, 2.8,
_Pv = {'ao': [-0.1567266, -5.539635, 0.7567212, -3.514322], 'exp': [0.5, 1, 2.5, 4]}
```

classmethod _Liquid_Density(*T*)

Auxiliary equation for the density or saturated liquid

Parameters **T** (*float*) – Temperature [K]

Returns **rho** – Saturated liquid density [kg/m^3]

Return type float

static _visco(*rho*, *T*, *fase=None*)

Equation for the Viscosity

Parameters

- **rho** (*float*) – Density, [kg/m^3]
- **T** (*float*) – Temperature, [K]

Returns μ – Viscosity, [Pa·s]

Return type float

References

Lemmon, E.W., Jacobsen, R.T. Viscosity and Thermal Conductivity Equations for Nitrogen, Oxygen, Argon, and Air. Int. J. Thermophys. 25 (1) (2004) 21-69. http://dx.doi.org/10.1023/B:IJOT.0000022327.04529.f3

_thermo(*rho*, *T*, *fase=None*)

Equation for the thermal conductivity

Parameters

- **rho** (*float*) – Density, [kg/m^3]
- **T** (*float*) – Temperature, [K]
- **fase** (*dict*) – phase properties

Returns **k** – Thermal conductivity, [W/mK]

Return type float

References

Lemmon, E.W., Jacobsen, R.T. Viscosity and Thermal Conductivity Equations for Nitrogen, Oxygen, Argon, and Air. Int. J. Thermophys. 25 (1) (2004) 21-69. http://dx.doi.org/10.1023/B:IJOT.0000022327.04529.f3

class iapws.humidAir.**HumidAir**(***kwargs*)

Humid air class with complete functionality

Parameters

- **T** (*float*) – Temperature, [K]
- **P** (*float*) – Pressure, [MPa]
- **rho** (*float*) – Density, [kg/m^3]
- **v** (*float*) – Specific volume, [m^3/kg]
- **A** (*float*) – Mass fraction of dry air in humid air, [kg/kg]
- **xa** (*float*) – Mole fraction of dry air in humid air, [-]
- **W** (*float*) – Mass fraction of water in humid air, [kg/kg]
- **xw** (*float*) – Mole fraction of water in humid air, [-]
- **HR** (*float*) – Humidity ratio, Mass fraction of water in dry air, [kg/kg]

Notes

- It needs two incoming properties of T, P, rho.
- v as a alternate input parameter to rho
- For composition need one of A, xa, W, xw, HR.

The calculated instance has the following properties:

- P: Pressure, [MPa]
- T: Temperature, [K]
- g: Specific Gibbs free energy, [kJ/kg]
- a: Specific Helmholtz free energy, [kJ/kg]
- v: Specific volume, [m^3/kg]
- rho: Density, [kg/m^3]
- h: Specific enthalpy, [kJ/kg]
- u: Specific internal energy, [kJ/kg]
- s: Specific entropy, [kJ/kg·K]
- cp: Specific isobaric heat capacity, [kJ/kg·K]
- w: Speed of sound, [m/s]
- alfav: Isobaric cubic expansion coefficient, [1/K]
- betas: Isoentropic temperature-pressure coefficient, [-]
- xkappa: Isothermal Expansion Coefficient, [-]
- ks: Adiabatic Compressibility, [1/MPa]
- A: Mass fraction of dry air in humid air, [kg/kg]
- W: Mass fraction of water in humid air, [kg/kg]
- xa: Mole fraction of dry air, [-]
- xw: Mole fraction of water, [-]
- Pv: Partial pressure of water, [MPa]

- xa_sat: Mole fraction of dry air at saturation state, [-]
- mu: Relative chemical potential, [kJ/kg]
- muw: Chemical potential of water, [kJ/kg]
- M: Molar mass of humid air, [g/mol]
- HR: Humidity ratio, Mass fraction of water in dry air, [kg/kg]
- RH: Relative humidity, [-]

Attributes

calculable	Check if inputs are enough to define state

Methods

__call__(**kwargs)	Make instance callable to can add input parameter one to one
calculo()	Calculate procedure
derivative(z, x, y)	Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, rho, u, h, s, g, a

status = 0

msg = 'Undefined'

kwargs = {'A': None, 'HR': None, 'P': 0.0, 'T': 0.0, 'W': None, 'rho': 0.0, 'v': 0.0

calculable
Check if inputs are enough to define state

calculo()
Calculate procedure

derivative(*z*, *x*, *y*)
Wrapper derivative for custom derived properties where x, y, z can be: P, T, v, rho, u, h, s, g, a

_eq(*T*, *P*)
Procedure for calculate the composition in saturation state

Parameters

- **T** (*float*) – Temperature [K]
- **P** (*float*) – Pressure [MPa]

Returns **Asat** – Saturation mass fraction of dry air in humid air [kg/kg]

Return type float

_prop(*T*, *rho*, *fav*)
Thermodynamic properties of humid air

Parameters

- **T** (*float*) – Temperature, [K]
- **rho** (*float*) – Density, [kg/m^3]
- **fav** (*dict*) – dictionary with helmholtz energy and derivatives

Returns

prop –

Dictionary with thermodynamic properties of humid air:

- P: Pressure, [MPa]
- s: Specific entropy, [kJ/kgK]
- cp: Specific isobaric heat capacity, [kJ/kgK]
- h: Specific enthalpy, [kJ/kg]
- g: Specific gibbs energy, [kJ/kg]
- alfav: Thermal expansion coefficient, [1/K]
- betas: Isentropic T-P coefficient, [K/MPa]
- xkappa: Isothermal compressibility, [1/MPa]
- ks: Isentropic compressibility, [1/MPa]
- w: Speed of sound, [m/s]

Return type dict

References

IAPWS, Guideline on an Equation of State for Humid Air in Contact with Seawater and Ice, Consistent with the IAPWS Formulation 2008 for the Thermodynamic Properties of Seawater, Table 5, http://www.iapws.org/relguide/SeaAir.html

_coligative(*rho*, *A*, *fav*)

Miscelaneous properties of humid air

Parameters

- **rho** (*float*) – Density, [kg/m^3]
- **A** (*float*) – Mass fraction of dry air in humid air, [kg/kg]
- **fav** (*dict*) – dictionary with helmholtz energy and derivatives

Returns

prop –

Dictionary with calculated properties:

- mu: Relative chemical potential, [kJ/kg]
- muw: Chemical potential of water, [kJ/kg]
- M: Molar mass of humid air, [g/mol]
- HR: Humidity ratio, [-]
- xa: Mole fraction of dry air, [-]
- xw: Mole fraction of water, [-]

Return type dict

References

IAPWS, Guideline on an Equation of State for Humid Air in Contact with Seawater and Ice, Consistent with the IAPWS Formulation 2008 for the Thermodynamic Properties of Seawater, Table 12, http://www.iapws.org/relguide/SeaAir.html

_fav(*T*, *rho*, *A*)

Specific Helmholtz energy of humid air and derivatives

Parameters

- **T** (*float*) – Temperature, [K]
- **rho** (*float*) – Density, [kg/m^3]
- **A** (*float*) – Mass fraction of dry air in humid air, [kg/kg]

Returns

prop –

Dictionary with helmholtz energy and derivatives:

- fir, [kJ/kg]
- fira: $\frac{\partial f_{av}}{\partial A}\Big|_{T,\rho}$, [kJ/kg]
- firt: $\frac{\partial f_{av}}{\partial T}\Big|_{A,\rho}$, [kJ/kgK]
- ird: $\frac{\partial f_{av}}{\partial \rho}\Big|_{A,T}$, [kJ/m^{3}kg^2]
- firaa: $\frac{\partial^2 f_{av}}{\partial A^2}\Big|_{T,\rho}$, [kJ/kg]
- firat: $\frac{\partial^2 f_{av}}{\partial A \partial T}\Big|_{\rho}$, [kJ/kgK]
- firad: $\frac{\partial^2 f_{av}}{\partial A \partial \rho}\Big|_{T}$, [kJ/m^{3}kg^2]
- firtt: $\frac{\partial^2 f_{av}}{\partial T^2}\Big|_{A,\rho}$, [kJ/kgK^2]
- firdt: $\frac{\partial^2 f_{av}}{\partial \rho \partial T}\Big|_{A}$, [kJ/m^{3}kg^{2}K]
- firdd: $\frac{\partial^2 f_{av}}{\partial \rho^2}\Big|_{A,T}$, [kJ/m^{6}kg^3]

Return type dict

References

IAPWS, Guideline on an Equation of State for Humid Air in Contact with Seawater and Ice, Consistent with the IAPWS Formulation 2008 for the Thermodynamic Properties of Seawater, Table 6, http://www.iapws.org/relguide/SeaAir.html

_fmix(*T*, *rho*, *A*)

Specific Helmholtz energy of air-water interaction

Parameters

- **T** (*float*) – Temperature, [K]

- **rho** (*float*) – Density, [kg/m^3]
- **A** (*float*) – Mass fraction of dry air in humid air, [kg/kg]

Returns

prop –

Dictionary with helmholtz energy and derivatives:

- fir, [kJ/kg]
- fira: $\frac{\partial f_{mix}}{\partial A}\Big|_{T,\rho}$, [kJ/kg]
- firt: $\frac{\partial f_{mix}}{\partial T}\Big|_{A,\rho}$, [kJ/kgK]
- fird: $\frac{\partial f_{mix}}{\partial \rho}\Big|_{A,T}$, [kJ/m^{3}kg^2]
- firaa: $\frac{\partial^2 f_{mix}}{\partial A^2}\Big|_{T,\rho}$, [kJ/kg]
- firat: $\frac{\partial^2 f_{mix}}{\partial A \partial T}\Big|_{\rho}$, [kJ/kgK]
- firad: $\frac{\partial^2 f_{mix}}{\partial A \partial \rho}\Big|_{T}$, [kJ/m^{3}kg^2]
- firtt: $\frac{\partial^2 f_{mix}}{\partial T^2}\Big|_{A,\rho}$, [kJ/kgK^2]
- firdt: $\frac{\partial^2 f_{mix}}{\partial \rho \partial T}\Big|_{A}$, [kJ/m^{3}kg^{2}K]
- firdd: $\frac{\partial^2 f_{mix}}{\partial \rho^2}\Big|_{A,T}$, [kJ/m^{6}kg^3]

Return type dict

References

IAPWS, Guideline on an Equation of State for Humid Air in Contact with Seawater and Ice, Consistent with the IAPWS Formulation 2008 for the Thermodynamic Properties of Seawater, Table 10, http://www.iapws.org/relguide/SeaAir.html

For a rapid usage demostration, see this examples

IAPWS-IF97 (see full documentation)

```
from iapws import IAPWS97
sat_steam=IAPWS97(P=1,x=1)                #saturated steam with known P
sat_liquid=IAPWS97(T=370, x=0)            #saturated liquid with known T
steam=IAPWS97(P=2.5, T=500)               #steam with known P and T
print(sat_steam.h, sat_liquid.h, steam.h) #calculated enthalpies
```

IAPWS-95 (see full documentation)

```
from iapws import IAPWS95
sat_steam=IAPWS95(P=1,x=1)                #saturated steam with known P
sat_liquid=IAPWS95(T=370, x=0)            #saturated liquid with known T
steam=IAPWS95(P=2.5, T=500)               #steam with known P and T
print(sat_steam.h, sat_liquid.h, steam.h) #calculated enthalpies
```

IAPWS-17 for Heavy water (see full documentation)

```
from iapws import D2O
sat_liquid=D2O(T=370, x=0)              #saturated liquid with known T
print(sat_liquid.h)                     #calculated enthalpy
```

IAPWS-06 for Ice Ih (see full documentation)

```
from iapws import _Ice
ice=_Ice(273.15, 0.101325)              #Ice at normal melting point
print(ice["rho"])                       #Calculated density
```

IAPWS-08 for seawater (see full documentation)

```
from iapws import SeaWater
state = SeaWater(T=300, P=0.101325, S=0.001)    #Seawater with 0.1% Salinity
print(state.cp)                                 # Get cp
```

CHAPTER 6

TODO

- FIXME: Electrolytic conductiviy
- TODO: Improve convergence in two phase region for IAPWS95 and D2O class
- TODO: Implement SBTL method for fast calculation
- TODO: Implement TTSE method for fast calculation

Ammonia-water mixture:

- TODO: Add equilibrium routine

I've tried to test all code and use all values for computer verification the standards give, but anyway the code can have hidden problem. For any suggestions, comments, bugs ... you can usage the github issue section, or contact directly with me at email.

CHAPTER 7

Introduction

Python implementation of standard from IAPWS (http://www.iapws.org/release.html).

- Home: https://github.com/jjgomera/iapws
- Author: Juan José Gómez Romera <jjgomera@gmail.com>
- License: GPL-3
- Documentation: http://iapws.readthedocs.io/

CHAPTER 8

Dependences

- python 2x, 3x, compatible with both versions
- Numpy-scipy: library with mathematic and scientific tools

CHAPTER 9

Installation

In debian you can find in official repositories in jessie, testing and sid. In ubuntu it's in official repositories from ubuntu saucy (13.10). In other system you can install using pip:

```
pip install iapws
```

or directly cloning the github repository:

```
git clone https://github.com/jjgomera/iapws.git
```

and adding the folder to a python path.

CHAPTER 10

Features

This module implements almost the full set of standards:

Releases:

- R1-76(2014): Revised Release on the Surface Tension of Ordynary Water Substance, *iapws._iapws._Tension()*
- R2-83(1992): Release on the Values of Temperature, Pressure and Density of Ordynary and Heavy Water Substances at their Respectives Critical Points, *iapws._iapws()*
- R5-85(1994): Release on Surface Tension of Heavy Water Substance, *iapws._iapws._D2O_Tension()*
- R6-95(2018): Revised Release on the IAPWS Formulation 1995 for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use, *iapws.iapws95.IAPWS95()*
- R7-97(2012): Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, *iapws.iapws97()*
- R8-97: Release on the Static Dielectric Constant of Ordinary Water Substance for Temperatures from 238 K to 873 K and Pressures up to 1000 MPa, *iapws._iapws._Dielectric()*
- R9-97: Release on the Refractive Index of Ordinary Water Substance as a Function of Wavelength, Temperature and Pressure, *iapws._iapws._Refractive()*
- R10-06(2009): Revised Release on the Equation of State 2006 for H2O Ice Ih, *iapws._iapws._Ice()*
- R11-07(2019): Release on the Ionization Constant of H2O, *iapws._iapws._Kw()*
- R12-08: Release on the IAPWS Formulation 2008 for the Viscosity of Ordinary Water Substance, *iapws._iapws._Viscosity()*
- R13-08: Release on the IAPWS Formulation 2008 for the Thermodynamic Properties of Seawater, *iapws.iapws08()*
- R14-08(2011): Revised Release on the Pressure along the Melting and Sublimation Curves of Ordinary Water Substance, *iapws._iapws._Melting_Pressure()*, *iapws._iapws._Sublimation_Pressure()*

- R15-11: Release on the IAPWS Formulation 2011 for the Thermal Conductivity of Ordinary Water Substance, *iapws._iapws._ThCond()*
- R16-17(2018): Release on the IAPWS Formulation 2017 for the Thermodynamic Properties of Heavy Water, *iapws.iapws95.D2O()*
- R17-20: Release on the IAPWS Formulation 2020 for the Viscosity of Heavy Water, *iapws._iapws._D2O_Viscosity()*
- R18-21: Release on the IAPWS Formulation 2021 for the Thermal Conductivity of Heavy Water, *iapws._iapws._D2O_ThCond()*

Supplementary Releases:

- SR1-86(1992): Revised Supplementary Release on Saturation Properties of Ordinary Water Substance, *iapws.iapws95.MEoS._Liquid_Density()*, *iapws.iapws95.MEoS._Vapor_Density()*, *iapws.iapws95.MEoS._Vapor_Pressure()*
- SR2-01(2014): Revised Supplementary Release on Backward Equations for Pressure as a Function of Enthalpy and Entropy p(h,s) for Regions 1 and 2 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, *iapws.iapws97._Backward1_P_hs()*, *iapws.iapws97._Backward2_P_hs()*
- SR3-03(2014): Revised Supplementary Release on Backward Equations for the Functions T(p,h), v(p,h), and T(p,s), v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, *iapws.iapws97._Backward3_T_Ph()*, *iapws.iapws97._Backward3_T_Ps()*, *iapws.iapws97._Backward3_v_Ph()*, *iapws.iapws97._Backward3_v_Ps()*
- SR4-04(2014): Revised Supplementary Release on Backward Equations p(h,s) for Region 3, Equations as a Function of h and s for the Region Boundaries, and an Equation Tsat(h,s) for Region 4 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, *iapws.iapws97._Backward3_P_hs()*
- SR5-05(2016): Revised Supplementary Release on Backward Equations for Specific Volume as a Function of Pressure and Temperature v(p,T) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, *iapws.iapws97._Backward3_v_PT()*
- SR6-08(2011): Revised Supplementary Release on Properties of Liquid Water at 0.1 MPa, *iapws._iapws._Liquid()*
- SR7-09: Supplementary Release on a Computationally Efficient Thermodynamic Formulation for Liquid Water for Oceanographic Use, *iapws.iapws08.SeaWater._waterSupp()*

Guidelines:

- G1-90: Electrolytic Conductivity (Specific Conductance) of Liquid and Dense Supercritical Water from 0°C to 800°C and Pressures up to 1000 MPa, *iapws._iapws._Conductivity()*
- G2-90(1994): Solubility of Sodium Sulfate in Aqueous Mixtures of Sodium Chloride and Sulfuric Acid from Water to Concentrated Solutions, from 250 °C to 350 °C, *iapws.iapws08._solNa2SO4()*
- G3-00(2012): Revised Guideline on the Critical Locus of Aqueous Solutions of Sodium Chloride, *iapws.iapws08._critNaCl()*
- G4-01: Guideline on the IAPWS Formulation 2001 for the Thermodynamic Properties of Ammonia-Water Mixtures, *iapws.ammonia()*
- G5-01(2016): Guideline on the Use of Fundamental Physical Constants and Basic Constants of Water, *iapws._iapws()*
- G6-03: Guideline on the Tabular Taylor Series Expansion (TTSE) Method for Calculation of Thermodynamic Properties of Water and Steam Applied to IAPWS-95 as an Example (Not implemented)

- G7-04: Guideline on the Henry's Constant and Vapor-Liquid Distribution Constant for Gases in H2O and D2O at High Temperatures, *iapws._iapws._Henry()*, *iapws._iapws._Kvalue()*
- G8-10: Guideline on an Equation of State for Humid Air in Contact with Seawater and Ice, Consistent with the IAPWS Formulation 2008 for the Thermodynamic Properties of Seawater, *iapws.humidAir.HumidAir()*
- G9-12: Guideline on a Low-Temperature Extension of the IAPWS-95 Formulation for Water Vapor, *iapws.iapws95.IAPWS95._phiex()*
- G10-15: Guideline on the Thermal Conductivity of Seawater, *iapws.iapws08._ThCond_SeaWater()*
- G11-15: Guideline on a Virial Equation for the Fugacity of H2O in Humid Air, *iapws.humidAir._virial()*
- G12-15: Guideline on Thermodynamic Properties of Supercooled Water, *iapws._iapws._Supercooled()*
- G13-15: Guideline on the Fast Calculation of Steam and Water Properties with the Spline-Based Table Look-Up Method (SBTL) (Not implemented)
- G14-19: Guideline on the Surface Tension of Seawater, *iapws.iapws08._Tension_SeaWater()*

Advisory Notes:

- AN1-03: Uncertainties in Enthalpy for the IAPWS Formulation 1995 for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use (IAPWS-95) and the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam (IAPWS-IF97)
- AN2-04(2013): Role of Various IAPWS Documents Concerning the Thermodynamic Properties of Ordinary Water Substance
- AN3-07(2018): Thermodynamic Derivatives from IAPWS Formulations, *iapws._utils.deriv_G()*, *iapws._utils.deriv_H()*
- AN4-09: Roles of IAPWS and CIPM Standards for the Density of Water
- AN5-13(2016): Industrial Calculation of the Thermodynamic Properties of Seawater, `iapws.iapws08.Seawater._waterIF97()`, *iapws.iapws08._Tb()*, *iapws.iapws08._Tf()*, *iapws.iapws08._Triple()*, *iapws.iapws08._OsmoticPressure()*
- AN6-16: Relationship between Various IAPWS Documents and the International Thermodynamic Equation of Seawater - 2010 (TEOS-10)

CHAPTER 11

Documentation

You can navigate the full documentation of package:

For a rapid usage demostration, see this examples

IAPWS-IF97 (see full documentation)

```
from iapws import IAPWS97
sat_steam=IAPWS97(P=1,x=1)                #saturated steam with known P
sat_liquid=IAPWS97(T=370, x=0)            #saturated liquid with known T
steam=IAPWS97(P=2.5, T=500)               #steam with known P and T
print(sat_steam.h, sat_liquid.h, steam.h) #calculated enthalpies
```

IAPWS-95 (see full documentation)

```
from iapws import IAPWS95
sat_steam=IAPWS95(P=1,x=1)                #saturated steam with known P
sat_liquid=IAPWS95(T=370, x=0)            #saturated liquid with known T
steam=IAPWS95(P=2.5, T=500)               #steam with known P and T
print(sat_steam.h, sat_liquid.h, steam.h) #calculated enthalpies
```

IAPWS-17 for Heavy water (see full documentation)

```
from iapws import D2O
sat_liquid=D2O(T=370, x=0)            #saturated liquid with known T
print(sat_liquid.h)                   #calculated enthalpy
```

IAPWS-06 for Ice Ih (see full documentation)

```
from iapws import _Ice
ice=_Ice(273.15, 0.101325)            #Ice at normal melting point
print(ice["rho"])                     #Calculated density
```

IAPWS-08 for seawater (see full documentation)

```
from iapws import SeaWater
state = SeaWater(T=300, P=0.101325, S=0.001)     #Seawater with 0.1% Salinity
print(state.cp)                                  # Get cp
```

CHAPTER 12

TODO

- FIXME: Electrolytic conductiviy
- TODO: Improve convergence in two phase region for IAPWS95 and D2O class
- TODO: Implement SBTL method for fast calculation
- TODO: Implement TTSE method for fast calculation

Ammonia-water mixture:

- TODO: Add equilibrium routine

I've tried to test all code and use all values for computer verification the standards give, but anyway the code can have hidden problem. For any suggestions, comments, bugs ... you can usage the github issue section, or contact directly with me at email.

CHAPTER 13

Indices and tables

- genindex
- modindex
- search

Python Module Index

i

Index

Symbols

A

C

D

F

G

H

I

K

M

N

P

R

S

T