iapws

Table of Contents

[0%] iapws_version.f90

[10%] iapws_version.f90 done.

[10%] iapws_api.f90

[20%] iapws_api.f90 done.

[20%] iapws_capi.f90

[30%] iapws_capi.f90 done.

[30%] iapws.f90

[40%] iapws.f90 done.

[40%] libiapws.a

[50%] libiapws.a done.

[50%] main.f90

[60%] main.f90 done.

[60%] example.f90

[70%] example.f90 done.

[70%] iapws

[80%] iapws done.

[80%] example_in_c

[90%] example_in_c done.

[90%] example_in_f

[100%]

example_in_f done. [100%] Project compiled successfully.

NAME

iapws - Compute light and heavy water properties.

SYNOPSIS

iapws *SUBCOMMAND* [OPTION...]

DESCRIPTION

iapws is a command line interface for computing properties of light and heavy water according to IAPWS.

SUBCOMMANDS

Valid subcommands are:

+kh Compute the Henry's constant for gases in H2O or D2O. The default behavior is to compute the constant kH for O2 at 25Â°C. See options.

+kd Compute the vapor-liquid distribution constant for gases in H2O or D2 The default behavior is to compute the constant kD for H2 at 25Â°C. See options.

+psat Compute the saturation pressure. The default behavior is to compute psat at 25Â°C. See options.

+Tsat Compute the saturation temperature. The default behavior is to compute Tsat at 1 bar. See options.

Their syntax is:

+kh [OPTION...]

+kd [OPTION...]

+psat [OPTION...]

+Tsat [OPTION...]

OPTIONS

kh:

--temperature, -T TEMPERATURE...
Temperature in Â°C. Default to 25Â°C.

--fugacity, -f FUGACITY...
Liquid-phase fugacity in MPa. Default to 1 b

--gases, -g GAS...
Gases for which to compute kH. Default to O2

--D2O Set heavywater as the solvent.

--listgases
Display available gases for computing kH.

kd:

--temperature, -T TEMPERATURE...
Temperature in Â°C. Default to 25Â°C.

--x2, -x x2...
Molar fraction of gas in water. Default to 1

--gases, -g GAS...
Gases for which to compute kD. Default to H2

--D2O, Set heavywater as the solvent.

--listgases
Display available gases for computing kD.

psat:

--temperature, -T TEMPERATURE...
Temperature in Â°C. Default to 25Â°C.

Tsat:

--pressure, -p PRESSURE...
Pressure in bar. Default to 1 bar.

all:

--usage, -u
Show usage text and exit.

--help, -h
Show help text and exit.

--verbose, -V
Display additional information when availabl

--version, -v
Show version information and exit.

EXAMPLE

Minimal example

```
iapws kh -T 25,100 -f 1,0.2 -g O2,H2

iapws kd -T 25,100 -x2 1d-9,1d-6 -g O2,H2
```

SEE ALSO

ciaaw(3), **codata**(3)

NAME

iapws - library for light and heavy water properties according to IAPWS.

LIBRARY

iapws (**-libiapws**, **-libiapws**)

SYNOPSIS

```
use iapws
include "iapws.h"
import pyiapws
```

DESCRIPTION

Numerical implementation for reports:

- R2-83
  - [x] Tc in H2O and D2O
  - [x] pc in H2O and D2O
  - [x] rhoc in H2O and D2O
- G7-04
  - [x] kH
  - [x] kD
- R7-97
  - [x] Region 1
  - [ ] Region 2
  - [ ] Region 3
  - [x] Region 4
  - [ ] Region 5
- R11-24:
  - [x] Kw

Fortran API

o real(dp), parameter :: Tc_H2O = 647.096_dp
Critical temperature for H2O in K

o real(dp), parameter :: Tc_D2O = 643.847_dp
Critical temperature for D2O in K

o real(dp), parameter :: pc_H2O = 22.064_dp
Critical pressure for H2O in MPa

o real(dp), parameter :: pc_D2O = 21.671_dp
Critical pressure for H2O in MPa

o real(dp), parameter :: rhoc_H2O = 322.0_dp
Critical density for H2O in kg.m-3

o real(dp), parameter :: rhoc_D2O = 356.0_dp
Critical density for H2O in kg.m-3

o function get_version()result(fptr)
Return the version

o character(len=:), pointer :: fptr
Fortran pointer to a string indicating the version.

- **pure subroutine kh(T, gas, heavywater, k)**
  Compute the henry constant kH in MPa for a given temperature (x_2=1/kH).
  - **real(dp), intent(in), contiguous :: T(:)**
    Temperature in K.
  - **character(len=*), intent(in) :: gas**
    Gas.
  - **integer(int32), intent(in) :: heavywater**
    Flag if D2O (1) is used or **H2O**(0).
  - **real(dp), intent(out), contiguous :: k(:)**
    Henry constant in MPa. Filled with NaNs if gas not found.
- **pure subroutine kd(T, gas, heavywater, k)**
  Compute the vapor-liquid constant kd for a given temperature (kd=y_2/x_2).
  - **real(dp), intent(in), contiguous :: T(:)**
    Temperature in K.
  - **character(len=*), intent(in) :: gas**
    Gas.
  - **integer(int32), intent(in) :: heavywater**
    Flag if D2O (1) is used or **H2O**(0).
  - **real(dp), intent(out), contiguous :: k(:)**
    Vapor-liquid constant (adimensional). Filled with NaNs if gas not found.
- **pure function ngases(heavywater)result(n)**
  Returns the number of gases.
  - **integer(int32), intent(in) :: heavywater**
    Flag if D2O (1) is used or **H2O**(0).
  - **integer(int32) :: n**
    Number of gases.
- **function gases(heavywater)result(list_gases)**
  Returns the list of available gases.
  - **integer(int32), intent(in) :: heavywater**
    Flag if D2O (1) is used or **H2O**(0).
  - **type(gas_type), pointer :: list_gases(:)**
    Available gases.
- **function gases2(heavywater)result(str_gases)**
  Returns the available gases as a string.
  - **integer(int32), intent(in) :: heavywater**
    Flag if D2O (1) is used or **H2O**(0).
  - **character(len=:), pointer :: str_gases**
    Available gases
- **pure subroutine psat(Ts, ps)**
  Compute the saturation pressure at temperature Ts (273.13 K <= Ts <= 647.096 K).
  - **real(dp), intent(in), contiguous :: Ts(:)**
    Saturation temperature in K.
  - **real(dp), intent(out), contiguous :: ps(:)**
    Saturation pressure in MPa. Filled with nan if out of validity range.

- **pure subroutine Tsat(ps, Ts)**
  Compute the saturation temperature at pressure ps (611.213 Pa <= ps <= 22.064 MPa).
  - **real(dp), intent(in), contiguous :: ps(:)**
    Saturation pressure in MPa.
  - **real(dp), intent(out), contiguous :: Ts(:)**
    Saturation temperature in K. Filled with nan if out of validity range.
- **pure subroutine wp(p, T, prop, res)**
  Compute water properties at pressure p in MPa and temperature T in Kelvin.
  - **real(dp), intent(in) :: p(:)**
    Pressure in MPa.
  - **real(dp), intent(in) :: T(:)**
    Pressure in K.
  - **character(len=*), intent(in) :: prop**
    Property (v, u, s, h, cp, cv, w)
  - **real(dp), intent(out) :: res(:)**
    Filled with NaN if no adequate region is found.
- **pure subroutine wr(p, T, res)**
  Get the water region corresponding to p and T.
  - **real(dp), intent(in) :: p(:)**
    Pressure in MPa.
  - **real(dp), intent(in) :: T(:)**
    Temperature in K.
  - **integer(int32), intent(out) :: res(:)**
    Region 1 to 5 if found or **-1**.
- **pure subroutine wph(p, T, res)**
  Get the water phase corresponding to p and T.
  - **real(dp), intent(in) :: p(:)**
    pressure in MPa.
  - **real(dp), intent(in) :: T(:)**
    Temperature in K.
  - **character(len=1), intent(out) :: res(:)**
    Phases: **l**(liquid), **v**(VAPOR), **c**(SUPER CRITICAL), **s**(SATURATION), **n**(UNKNOWN).
- **pure subroutine Kw(T, rhow, k)**
  Compute the ionization constant of water Kw (273.13 K <= T <= 1273.15 K and 0 <= p <= 1000 MPa).
  - **real(dp), intent(in) :: T(:)**
    Temperature in K.
  - **real(dp), intent(in) :: rhow(:)**
    Mass density in g.cm^{**-3**}.
  - **real(dp), intent(out) :: k(:)**
    Ionization constant. Filled with NaN if out of validity range.

C API

- char* **iapws_get_version**(void)

- const double iapws_r283_Tc_H2O
- const double iapws_r283_Tc_D2O
- const double iapws_r283_pc_H2O
- const double iapws_r283_pc_D2O
- const double iapws_r283_rhoc_H2O
- const double iapws_r283_rhoc_D2O
- void **iapws_g704_kh**(double *T, char *gas, int heavywater, double *k, int size_gas, size_t size_T)
- void **iapws_g704_kd**(double *T, char *gas, int heavywater, double *k, int size_gas, size_t size_T)
- int **iapws_g704_ngases**(int heavywater)
- char **iapws_g704_gases(int heavywater)
- char ***iapws_g704_gases2**(int heavywater)
- void **iapws_r797_psat**(size_t N, double *Ts, double *ps)
- void **iapws_r797_Tsat**(size_t N, double *ps, double *Ts)
- void **iapws_r797_wp**(double *p, double *T, char *prop, double *res, size_t N, size_t len)
- void **iapws_r797_wr**(double *p, double *T, int *res, size_t N)
- void **iapws_r797_wph**(double *p, double *T, char *res, size_t N)
- void **iapws_r1124_Kw**(size_t N, double *T, double *rhow, double *k)

Python wrapper

- **kh**(T: np.ndarray, gas: str, heavywater: bool=False)->Union[np.ndarray, float]
- **kd**(T: np.ndarray, gas: str, heavywater: bool=False)->Union[np.ndarray, float]
- **ngases**(heavywater:bool=False)->int
- **gases**(heavywater: bool=False)->List[str]
- **gases2**(heavywater: bool=False)->str
- **psat**(Ts)->Union[np.ndarray, float]
- **Tsat**(ps)->Union[np.ndarray, float]
- **wp**(p, T, prop)->Union[np.ndarray, float]
- **wr**(p, T)->Union[np.ndarray, float]
- **wph**(p, T)->Union[np.ndarray, str]
- **Kw**(T: np.ndarray, rhow: np.ndarray)->Union[np.ndarray, float]

NOTES

To **use** *iapws* within your fpm project, add the following to your fpm.toml file:

```
[dependencies]
iapws = { git="https://github.com/MilanSkocic/iapws.git" }
```

- dp stands for double precision and it is an alias to real64 from the iso_fortran_env module.
- l => liquid
- v => vapor

- c => super critical
- s => saturation
- n => unknown

EXAMPLE

Example in Fortran

```
program example_in_f
use stdlib_kinds, only: dp, int32
use iapws
implicit none
integer(int32) :: i, ngas
real(dp) :: T(1), kh_res(1), kd_res(1), wp_res(1), p(1)
real(dp) :: Ts(7), ps(7)
real(dp) :: x(3), y(3)
integer(int32) :: r(3)
character(len=1) :: s(3)
character(len=2) :: gas = "O2"
integer(int32) :: heavywater = 0
type(gas_type), pointer :: gases_list(:)
character(len=:), pointer :: gases_str

print *, '########################## IAPWS VERSION ######################
print *, "version ", get_version()

print *, '########################## IAPWS R2-83 ########################
print "(a, f10.3, a)", "Tc in h2o=", Tc_H2O, " k"
print "(a, f10.3, a)", "pc in h2o=", pc_H2O, " mpa"
print "(a, f10.3, a)", "rhoc in h2o=", rhoc_H2O, " kg/m3"

print "(a, f10.3, a)", "Tc in D2O=", Tc_D2O, " k"
print "(a, f10.3, a)", "pc in D2O=", pc_D2O, " mpa"
print "(a, f10.3, a)", "rhoc in D2O=", rhoc_D2O, " kg/m3"
print *, ''

print *, '########################## IAPWS G7-04 ########################
! Compute kh and kd in H2O
T(1) = 25.0_dp + 273.15_dp
call kh(T, gas, heavywater, kh_res)
print "(A10, 1X, A10, 1X, A2, F10.1, A, 4X, A3, SP, F10.4)", "Gas=", gas,

call kd(T, gas, heavywater, kd_res)
print "(A10, 1X, A10, 1X, A2, F10.1, A, 4X, A3, SP, F15.4)", "Gas=", gas,

! Get and print available gases
heavywater = 0
ngas = ngases(heavywater)
gases_list => null()
gases_list => gases(heavywater)
gases_str => gases2(heavywater)
print *, "Gases in H2O: ", ngas
print *, gases_str
do i=1, ngas
    print *, gases_list(i)%gas
enddo
```

```
heavywater = 1
ngas = ngases(heavywater)
gases_list => null()
gases_list => gases(heavywater)
gases_str => gases2(heavywater)
print *, "Gases in D2O: ", ngas
print *, gases_str
do i=1, ngas
    print *, gases_list(i)%gas
enddo

print *, '########################## IAPWS R7-97 ########################
! Compute ps from Ts.
Ts(:) = [-1.0_dp, 25.0_dp, 100.0_dp, 200.0_dp, 300.0_dp, 360.0_dp, 374.0_
Ts(:) = Ts(:) + 273.15_dp
call psat(Ts, ps)

do i=1, size(Ts)
    print "(SP, F23.3, A3, 4X, F23.3, A3)", Ts(i), "K", ps(i), "MPa"
end do

! Compute Ts from ps
call Tsat(ps, Ts)
do i=1, size(Ts)
    print "(SP, F23.3, A3, 4X, F23.3, A3)", Ts(i), "K", ps(i), "MPa"
end do

! Compute water properties at 280Â°C/8 Mpa
p(1) = 8.0_dp
T(1) = 273.15_dp + 280.0_dp
call wp(p, T, "v", wp_res)
print "(A5, F23.16, X, A)", "v(8MPa,280Â°C)=", wp_res(1)*1000.0_dp, "L/kg

! Compute region and phase
x = [8.0_dp, 4.0_dp, 6.0_dp ]
y = [553.15_dp, 1200.0_dp, 2000.0_dp]
call wr(x, y, r)
call wph(x, y, s)
print *, r
print *, s

end program
```

Example in C

```
#include <string.h>
#include <stdio.h>
#include "iapws.h"

int main(void){
double T = 25.0 + 273.15; /* in C*/
double p; /* p in Mpa */
char *gas = "O2";
double kh, kd, wp_res;
char **gases_list;
```

```
char *gases_str;
int ngas;
int i;
int heavywater = 0;
double x[3]= {8.0, 4.0, 6.0 };
double y[3] = {553.15, 1200.0, 2000.0};
int r[3];
char s[3];

printf("%s0, "########################## IAPWS VERSION ##################
printf("version %s0, iapws_get_version());

printf("%s0, "########################## IAPWS R2-83 ####################
printf("%s %10.3f %s0, "Tc in H2O", iapws_r283_Tc_H2O, "K");
printf("%s %10.3f %s0, "pc in H2O", iapws_r283_pc_H2O, "MPa");
printf("%s %10.3f %s0, "rhoc in H2O", iapws_r283_rhoc_H2O, "kg/m3");

printf("%s %10.3f %s0, "Tc in D2O", iapws_r283_Tc_D2O, "K");
printf("%s %10.3f %s0, "pc in D2O", iapws_r283_pc_D2O, "MPa");
printf("%s %10.3f %s0, "rhoc in D2O", iapws_r283_rhoc_D2O, "kg/m3");

printf("0);


printf("%s0, "########################## IAPWS G7-04 ####################
/* Compute kh and kd in H2O*/
iapws_g704_kh(&T, gas, heavywater, &kh, strlen(gas), 1);
printf("Gas=%sT=%fKkh=%+10.4f0, gas, T, kh);

iapws_g704_kd(&T, gas, heavywater, &kd, strlen(gas), 1);
printf("Gas=%sT=%fKkd=%+15.4f0, gas, T, kd);

/* Get and print the available gases */
ngas = iapws_g704_ngases(heavywater);
gases_list = iapws_g704_gases(heavywater);
gases_str = iapws_g704_gases2(heavywater);
printf("Gases in H2O: %d0, ngas);
printf("%s0, gases_str);
for(i=0; i<ngas; i++){
    printf("%s0, gases_list[i]);
}

heavywater = 1;
ngas = iapws_g704_ngases(heavywater);
gases_list = iapws_g704_gases(heavywater);
gases_str = iapws_g704_gases2(heavywater);
printf("Gases in D2O: %d0, ngas);
printf("%s0, gases_str);
for(i=0; i<ngas; i++){
    printf("%s0, gases_list[i]);
}

printf("%s0, "########################## IAPWS R7-97 ####################
double Ts[7] =  {-1.0, 25.0, 100.0, 200.0, 300.0, 360.0, 374.0};
```

```
double ps[7] = {1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0};
for(i=0; i<7; i++){
    Ts[i] = Ts[i] + 273.15;
}
iapws_r797_psat(7, Ts, ps);

for(i=0; i<7; i++){
    printf("%+23.3f %s %+23.3f %s0, Ts[i], "K", ps[i], "MPa");
}

iapws_r797_Tsat(7, ps, Ts);
for(i=0; i<7; i++){
    printf("%+23.3f %s %+23.3f %s0, Ts[i], "K", ps[i], "MPa");
}

T = 273.15 + 280.0;
p = 8.0;
iapws_r797_wp(&p, &T, "v", &wp_res, 1, 1);
printf("v(8MPa,280Â°C) = %+23.16f L/kg0, wp_res * 1000.0);

iapws_r797_wr(x, y, r, 3);
iapws_r797_wph(x, y, s, 3);
for(i=0; i<3; i++){
    printf("%i", r[i]);
}
printf("0);
for(i=0; i<3; i++){
    printf("%c", s[i]);
}
printf("0);

return 0;
}
```

Example in Python

```
import array
import numpy as np
import matplotlib.pyplot as plt
import pyiapws

print("########################## IAPWS VERSION ##########################
print(pyiapws.__version__)

print("########################## IAPWS R2-83 ##########################"
print("Tc in H2O", pyiapws.Tc_H2O, "K")
print("pc in H2O", pyiapws.pc_H2O, "MPa")
print("rhoc in H2O", pyiapws.rhoc_H2O, "kg/m3")

print("Tc in D2O", pyiapws.Tc_D2O, "K")
print("pc in D2O", pyiapws.pc_D2O, "MPa")
print("rhoc in D2O", pyiapws.rhoc_D2O, "kg/m3")

print("")
```

```
print("########################## IAPWS G7-04 ##########################"
gas  = "O2"
T = array.array("d", (25.0+273.15,))

# Compute kh and kd in H2O
heavywater = False
k = pyiapws.kh(T, "O2", heavywater)
print(f"Gas={gas}T={T[0]}Kkh={k[0]:+10.4f}0)

k = pyiapws.kd(T, "O2", heavywater)
print(f"Gas={gas}T={T[0]}Kkd={k[0]:+10.4f}0)

# Get and print the available gases
heavywater = False
gases_list = pyiapws.gases(heavywater)
gases_str = pyiapws.gases2(heavywater)
ngas = pyiapws.ngases(heavywater)
print(f"Gases in H2O: {ngas:}")
print(gases_str)
for gas in gases_list:
    print(gas)

heavywater = True
gases_list = pyiapws.gases(heavywater)
gases_str = pyiapws.gases2(heavywater)
ngas = pyiapws.ngases(heavywater)
print(f"Gases in D2O: {ngas:}")
print(gases_str)
for gas in gases_list:
    print(gas)

style = {"marker":".", "ls":"", "ms":2}
T_KELVIN = 273.15
T = np.linspace(0.0, 360.0, 1000) + 273.15

solvent = {True: "D2O", False: "H2O"}

print("Generating plot for kh")
kname = "kh"
for HEAVYWATER in (False, True):
    print(solvent[HEAVYWATER])
    fig = plt.figure()
    ax = fig.add_subplot()
    ax.grid(visible=True, ls=':')
    ax.set_xlabel("T /Â°K")
    ax.set_ylabel("ln (kh/1GPa)")
    gases = pyiapws.gases(HEAVYWATER)
    for gas in gases:
        k = pyiapws.kh(T, gas, HEAVYWATER) / 1000.0
        ln_k = np.log(k)
        ax.plot(T, ln_k, label=gas, **style)
    ax.legend(ncol=3)
    fig.savefig(f"../media/g704-{kname:s}_{solvent[HEAVYWATER]}.png", dpi
```

```
print("Generating plot for kd")
kname = "kd"
for HEAVYWATER in (False, True):
    print(solvent[HEAVYWATER])
    fig = plt.figure()
    ax = fig.add_subplot()
    ax.grid(visible=True, ls=':')
    ax.set_xlabel("T /Â°K")
    ax.set_ylabel("ln kd")
    gases = pyiapws.gases(HEAVYWATER)
    for gas in gases:
        k = pyiapws.kd(T, gas, HEAVYWATER)
        ln_k = np.log(k)
        ax.plot(T, ln_k, label=gas, **style)
    ax.legend(ncol=3)
    fig.savefig(f"../media/g704-{kname:s}_{solvent[HEAVYWATER]}.png", dpi=


print("########################## IAPWS R7-97 ##########################"
Ts = np.asarray([-1.0, 25.0, 100.0, 200.0, 300.0, 360.0, 374.0])
Ts = Ts + 273.15


ps = pyiapws.psat(Ts)
for i in range(Ts.size):
    print(f"{Ts[i]:23.3f} K {ps[i]:23.3f} MPa.")

Ts = pyiapws.Tsat(ps)
for i in range(Ts.size):
    print(f"{Ts[i]:23.3f} K {ps[i]:23.3f} MPa.")

fig = plt.figure()
ax = fig.add_subplot()
ax.grid(visible=True, ls=':')
ax.set_xlabel("Ts /K")
ax.set_ylabel("ps /MPa")
Ts = np.linspace(0.0, 370.0, 500)
Ts = Ts + 273.15

ps = pyiapws.psat(Ts)
ax.plot(Ts, ps, "r-", label="ps(Ts)")

Ts = pyiapws.Tsat(ps)
ax.plot(Ts, ps, "b--", label="Ts(ps)")

ax.legend()
fig.savefig(f"../media/r797-r4.png", dpi=100, format="png")


T = 280.0 + 273.15
p = 8.0
res = pyiapws.wp(p, T, "v")*1000.0
print(f"v(8MPa,280Â°C) = {res:+23.16f} L/kg)")
```

```
x = np.asarray([8.0, 4.0, 6.0 ])
y = np.asarray([553.15, 1200.0, 2000.0])
r=pyiapws.wr(x, y)
s=pyiapws.wph(x, y)
print(r)
print(s)


plt.show()
```

SEE ALSO

codata(3), **ciaaw**(3)