ciaaw

Table of Contents

NAME

ciaaw - library for isotopic abundances and standard atomic weights

LIBRARY

ciaaw (**-libciaaw**, **-lciaaw**)

SYNOPSIS

```
use ciaaw
include "ciaaw.h"
import pyciaaw
```

DESCRIPTION

ciaaw is a Fortran library providing the standard and abridged atomic weights, the isotopic abundance and the isotopesâ standard atomic weights. The data are taken from http://ciaaw.org. C API allows usage from C, or can be used as a basis for other wrappers. Python wrapper allows easy usage from Python.

What have been implemented:

- **o SAW** Standard Atomic Weights.
- **o ICE** Isotopic Composition of the Element
- **o NAW** Nuclides Atomic Weight.

The latest standard atomic weights were released in 2021 by the *ciaaw* (https://www.ciaaw.org). All the values for the atomic weights are provided as double precision reals. The standard atomic weights (or realtive atomic mass), **A_r**(E), are extracted from table 1 from Prohaska et al. (IUPAC Technical Report **94**(5):573â600). For the elements that feature an interval for the standard atomic weight, the mean value and the uncertainty are computed using formulas defined in [2]. The standard atomic weights are a dimensionless quantity and thus they need to be multiplied by the molar mass constant Mu in order to get the value in g.mol-1. See codata for physical constants.

The latest isotopic compositions were released in 2013 by the *ciaaw* (https://www.ciaaw.org). All the values for the compositions are provided as double precision reals. The isotopic compositions of the element, are extracted from table 1 from Meija et al. (IUPAC Technical Report. **88**(3):293â306).

The latest atomic weights for nuclides were released in 2020 by *ciaaw* https://www.ciaaw.org from Huang et al. (AME 2020 **45**(3):030002). All the values for the nuclide atomic weights are provided as double precision reals.

The C API is defined by adding a prefix to the functions from the Fortran API due to the lack of module/namespace feature in the C language. The functions are therefore following this template: (c_prefix)fortran_func.

- (ciaaw_)get_saw
- (ciaaw_)get_ice
- (ciaaw_)get_nice
- (ciaaw_)get_naw
- (ciaaw_)get_nnaw

NOTES

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

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

EXAMPLES

Example in Fortran:

```
program example_in_f
```

```
use ciaaw
implicit none

character(len=8) :: s

! ASAW = Abridged Standard Atomic Weight
! SAW  = Standard Atomic Weight
! ICE  = Isotopic Composition of the Element
! NAW  = Nuclide Atomic Weight
! U    = Uncertainty

print '(A)', '########### CIAAW VERSION ##########'
print *, "version ", get_version()

print '(A)', '########### CIAAW SAW ##########'
print '(A10, F10.5)', 'ASAW H   = ', get_saw("H", abridged=.true.)
print '(A10, F10.5)', 'U ASAW H = ', get_saw("H", uncertainty=.true.)
print '(A10, F10.5)', 'SAW H    = ', get_saw("H", abridged = .false.)
print '(A10, F10.5)', 'U SAW H  =  ', get_saw("H", abridged = .false., un
print '(A10, F10.5)', 'ASAW T   = ', get_saw("Tc", abridged=.true.)

print '(A)', '########### CIAAW ICE ##########'
print '(A, I3)',        'N ICE H    = ', get_nice("H")
print '(A, F12.6)',   'ICE H 1    = ', get_ice("H", A=1)
print '(A, ES23.16)', 'U ICE H 1  = ', get_ice("H", A=1, uncertainty=.tru
print '(A, F12.6)',   'ICE H 2    = ', get_ice("H", A=2)
print '(A, ES23.16)', 'U ICE H 2  = ', get_ice("H", A=2, uncertainty=.tru
print '(A, I3)',        'N ICE Tc   = ', get_nice("Tc")
print '(A, I3)',        'N ICE C    = ', get_nice("C")

print '(A)', '########### CIAAW NAW ##########'
print '(A, ES23.16)',   'NAW H 2     = ', get_naw("H", A=2)
print '(A, ES23.16)',   'U NAW H 2   = ', get_naw("H", A=2, uncertainty=.
print '(A, I3)',          'N NAW Tc    = ', get_nnaw("Tc")
end program
```

Example in C:

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include "ciaaw.h"


// ASAW = Abridged Standard Atomic Weight
// SAW  = Standard Atomic Weight
// ICE  = Isotopic Composition of the Element
// NAW  = Nuclide Atomic Weight
// U    = Uncertainty

int main(void){
    printf("%s0, "########## CIAAW VERSION ##########");
    printf("version %s0, ciaaw_get_version());
```

```
printf("%s0, "########## CIAAW SAW ##########");
printf("%s %10.5f0, "ASAW H   = ", ciaaw_get_saw("H", 1, true, false)
printf("%s %10.5f0, "U ASAW H = ", ciaaw_get_saw("H", 1, true, true))
printf("%s %10.5f0, "SAW H    = ", ciaaw_get_saw("H", 1, false, false
printf("%s %10.5f0, "U SAW H  = ", ciaaw_get_saw("H", 1, false, true)
printf("%s %10.5f0, "ASAW Tc  = ", ciaaw_get_saw("Tc", 2, true, false

printf("%s0, "########## CIAAW ICE ##########");
printf("%s %d0,     "N ICE H      = ", ciaaw_get_nice("H", 1));
printf("%s %12.6f0, "ICE H 1      = ", ciaaw_get_ice("H", 1, 1, false
printf("%s %23.16e0,"U ICE H 1    = ", ciaaw_get_ice("H", 1, 1, true)
printf("%s %12.6f0, "ICE H 2      = ", ciaaw_get_ice("H", 1, 2, false
printf("%s %23.16e0,"U ICE H 2    = ", ciaaw_get_ice("H", 1, 2, true)
printf("%s %d0,     "N ICE Tc     = ", ciaaw_get_nice("Tc", 2));
printf("%s %d0,     "N ICE C      = ", ciaaw_get_nice("C", 1));

printf("%s0, "########## CIAAW NAW ##########");
printf("%s %23.16f0, "NAW H 2      = ", ciaaw_get_naw("H", 1, 2, fals
printf("%s %23.16e0, "U NAW H 2    = ", ciaaw_get_naw("H", 1, 2, true
printf("%s %d0,      "N NAW Tc     = ", ciaaw_get_nnaw("Tc", 2));
return EXIT_SUCCESS;
}
```

Example in Python:

```
import pyciaaw

# ASAW = Abridged Standard Atomic Weight
# SAW  = Standard Atomic Weight
# ICE  = Isotopic Composition of the Element
# NAW  = Nuclide Atomic Weight
# U    = Uncertainty

print("########## CIAAW VERSION ##########")
print("version ", pyciaaw.__version__)

print("########## CIAAW SAW  ##########")
print("ASAW H   = ", pyciaaw.get_saw("H"))
print("U ASAW H = ", pyciaaw.get_saw("H", uncertainty=True))
print("SAW H    = ", pyciaaw.get_saw("H", abridged=False, uncertainty=Fal
print("U SAW H  = ", pyciaaw.get_saw("H", abridged=False, uncertainty=Tru
print("ASAW Tc  = ", pyciaaw.get_saw("Tc"))


print("########## CIAAW ICE  ##########")
print("N ICE H   = ", pyciaaw.get_nice("H"))
print('ICE H 1   = ', pyciaaw.get_ice("H", A=1))
print('U ICE H 1 = ', pyciaaw.get_ice("H", A=1, uncertainty=True))
print('ICE H 2   = ', pyciaaw.get_ice("H", A=2))
print('U ICE H 2 = ', pyciaaw.get_ice("H", A=2, uncertainty=True))
print("N ICE Tc  = ", pyciaaw.get_nice("Tc"))
print("N ICE C   = ", pyciaaw.get_nice("C"))

print("########## CIAAW NAW  ##########")
print('NAW H 2   = ', pyciaaw.get_naw("H", A=2))
```

```
print('U NAW H 2 = ', pyciaaw.get_naw("H", A=2, uncertainty=True))
print("N NAW Tc  = ", pyciaaw.get_nnaw("Tc"))
```

SEE ALSO

gsl(3), **codata**(3), **ciaaw_get_version**(3), **ciaaw_get_saw**(3) **ciaaw_get_ice**(3), **ciaaw_get_nice**(3), **ciaaw_get_naw**(3), **ciaaw_get_naw**(3)

NAME

get_version - version getter for the library

LIBRARY

ciaaw - (**-libciaaw**, **-lciaaw**)

SYNOPSIS

```
function get_version()result(fptr)
```

DESCRIPTION

This function returns the version of the ciaaw library.

RETURN VALUE

character(len=:), pointer :: *fptr*

NOTES

The C API is defined by the following prototype: char* **ciaaw_get_version**(void)

The python wrapper embeds the version number in the top level variable __version__.

EXAMPLE

Fortran

```
print *, "version = ", get_version()
```

C

```
printf("version = %s0, ciaaw_get_version());
```

Python

```
print(f"version = {pyciaaw.__version__}")
```

SEE ALSO

ciaaw(3)

NAME

saw - get the standard atomic weight

LIBRARY

ciaaw - (**-libciaaw**, **-lciaaw**)

SYNOPSIS

```
function get_saw(s, abridged, uncertainty)result(res)
```

DESCRIPTION

This function returns the standard atomic weight.

Parameters:

o character(len=*), intent(in) :: *s*
Element symbol.

o logical, intent(in), optional :: *abridged*
Flag for returning the *abridged* standard atomic weight. Default to TRUE.

o logical, intent(in), optional :: uncertainty
Flag for returning the uncertainty instead of the value. Default to FALSE.

RETURN VALUE

real(dp) :: *res*

NOTES

The C API is defined by the following prototype:

double **ciaaw_get_saw**(char **s*, int n, bool *abridged*, bool uncertainty)

The python wrapper is defined by the following prototype:

def **get_saw**(*s*: str, *abridged*: bool=True, uncertainty: bool=False)->float

EXAMPLE

Fortran

```
print '(A10, F10.5)', 'ASAW H   = ', get_saw("H")
```

C

```
printf("%s %10.5f0, "ASAW H   = ", ciaaw_get_saw("H", 1, true, false));
```

Python

```
print("ASAW H   = ", pyciaaw.get_saw("H"))
```

SEE ALSO

ciaaw(3)