ciaaw Documentation

Release 0.1.0

M. Skocic

Feb 28, 2024

CONTENTS:

CHAPTER

ONE

GETTING STARTED

Sources: https://github.com/MilanSkocic/ciaaw

1.1 ciaaw

ciaaw is a Fortran library providing the standard and abreged atomic weights, the isotopic abundance and the isotopes standard atomic weights. It also provides a API for the C language. The formulas are taken from http://ciaaw.org.

1.1.1 How to install

A Makefile is provided, which uses fpm, for building the library.

On windows, msys2 needs to be installed. The MSVC compiler is only necessary for compiling the python wrapper. Add the msys2 binary (usually C:\msys64\usr\bin) to the path in order to be able to use make.

On Darwin, the gcc toolchain needs to be installed.

Build: the configuration file will set all the environmental variables necessary for the compilation

```
chmod +x configure.sh
. ./configure.sh
make
```

Run tests

```
make test
```

Install

```
make install
```

Uninstall

```
make uninstall
```

If building the python wrapper is needed:

```
cd pywrapper
chmod +x configure.sh
. ./configure.sh
make
```

1.1.2 Dependencies

```
gcc>=10.0
gfortran>=10.0
fpm>=0.7
```

1.1.3 License

GNU General Public License v3 (GPLv3)

1.2 pyciaaw

Python wrapper around the Fortran ciaaw library. The Fortran library does not need to be installed, the python wrapper embeds all needed fortran dependencies. On linux, you might have to install *libgfortran* if it is not distributed by default with your linux distribution.

1.2.1 How to install

```
pip install pyciaaw
```

1.2.2 Dependencies

```

```

1.2.3 License

GNU General Public License v3 (GPLv3)

1.3 Examples

1.3.1 Example in Fortran

```
program example_in_f
    use ciaaw
    implicit none

    type(element_t) :: elmt

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

    print '(A)', '########### CIAAW SAW ##########'
    elmt = H
    print '(A, A)', 'Element: ', elmt%element
    print '(A, A)', 'Symbol: ', elmt%symbol
    print '(A, I3)', 'Z: ', elmt%z
    print '(A, F23.16)', 'standard atomic weight max: ', elmt%saw_max
```

(continues on next page)

(continued from previous page)

```
    print '(A, F23.16)', 'standard atomic weight min: ', elmt%saw_min
    print '(A, F23.16)', 'standard atomic weight: ', elmt%saw
    print '(A, F23.16)', 'standard atomic weight uncertainty: ', elmt%saw_u
    print '(A, F23.16)', 'abredged standard atomic weight: ', elmt%asaw
    print '(A, F23.16)', 'abredged standard atomic weight uncertainty: ', elmt%asaw_u
    print '(A)', ''
    elmt = F
    print '(A, A)', 'Element: ', elmt%element
    print '(A, A)', 'Symbol: ', elmt%symbol
    print '(A, I3)', 'Z: ', elmt%z
    print '(A, F23.16)', 'standard atomic weight max: ', elmt%saw_max
    print '(A, F23.16)', 'standard atomic weight min: ', elmt%saw_min
    print '(A, F23.16)', 'standard atomic weight: ', elmt%saw
    print '(A, F23.16)', 'standard atomic weight uncertainty: ', elmt%saw_u
    print '(A, F23.16)', 'abredged standard atomic weight: ', elmt%asaw
    print '(A, F23.16)', 'abredged standard atomic weight uncertainty: ', elmt%asaw_u

end program
```

1.3.2 Example in C

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


int main(void){

    struct ciaaw_saw_element_t elmt;

    printf("%s\n", "########## CIAAW VERSION ##########");
    printf("version %s\n", ciaaw_get_version());

    printf("%s\n", "########## CIAAW SAW ##########");
    elmt = ciaaw_saw_H;
    printf("%s %s\n", "Element: ", elmt.element);
    printf("%s %s\n", "Symbol: ", elmt.symbol);
    printf("%s %d", "Z: ", elmt.z);
    printf("%s %23.16f\n", "standard atomic weight max: ", elmt.saw_max);
    printf("%s %23.16f\n", "standard atomic weight min: ", elmt.saw_min);
    printf("%s %23.16f\n", "standard atomic weight: ", elmt.saw);
    printf("%s %23.16f\n", "standard atomic weight uncertainty: ", elmt.saw_u);
    printf("%s %23.16f\n", "abredged standard atomic weight: ", elmt.asaw);
    printf("%s %23.16f\n", "abredged standard atomic weight uncertainty: ", elmt.asaw_
↪u);
    printf("%s\n", "");

    elmt = ciaaw_saw_F;
    printf("%s %s\n", "Element: ", elmt.element);
    printf("%s %s\n", "Symbol: ", elmt.symbol);
    printf("%s %d", "Z: ", elmt.z);
    printf("%s %23.16f\n", "standard atomic weight max: ", elmt.saw_max);
    printf("%s %23.16f\n", "standard atomic weight min: ", elmt.saw_min);
    printf("%s %23.16f\n", "standard atomic weight: ", elmt.saw);
```

(continues on next page)

(continued from previous page)

```
    printf("%s %23.16f\n", "standard atomic weight uncertainty: ", elmt.saw_u);
    printf("%s %23.16f\n", "abredged standard atomic weight: ", elmt.asaw);
    printf("%s %23.16f\n", "abredged standard atomic weight uncertainty: ", elmt.asaw_
↪u);

    return EXIT_SUCCESS;
}
```

1.3.3 Example in Python

```
import pyciaaw

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

print("########## CIAAW SAW ##########")
elmt = pyciaaw.saw.H
print("Element: ", elmt["element"])
print("Symbol: ", elmt["symbol"])
print("Z: ", elmt["z"])
print("standard atomic weight max: ", elmt["saw_max"])
print("standard atomic weight min: ", elmt["saw_min"])
print("standard atomic weight: ", elmt["saw"])
print("standard atomic weight uncertainty: ", elmt["saw_u"])
print("abredged standard atomic weight: ", elmt["asaw"])
print("abredged standard atomic weight uncertainty: ", elmt["asaw_u"])
print()

elmt = pyciaaw.saw.F
print("Element: ", elmt["element"])
print("Symbol: ", elmt["symbol"])
print("Z: ", elmt["z"])
print("standard atomic weight max: ", elmt["saw_max"])
print("standard atomic weight min: ", elmt["saw_min"])
print("standard atomic weight: ", elmt["saw"])
print("standard atomic weight uncertainty: ", elmt["saw_u"])
print("abredged standard atomic weight: ", elmt["asaw"])
print("abredged standard atomic weight uncertainty: ", elmt["asaw_u"])
```

CHAPTER

TWO

USER GUIDE

2.1 Standard Atomic Weights

The standard atomic weights (or realtive atomic mass), $A_r(E)$, are extracted from table 1 in Prohaska *et al.* [1]. For the elements that feature an interval for the standard atomic weight, the mean value and the uncertainty are computed using formulas defined in van der Veen *et al.* [2]:

$$A_r(E) = \frac{a+b}{2}$$
$$u(A_r(E)) = \frac{b-a}{2\sqrt{3}}$$

The standard atomic weights are a dimensionless quantity and thus they need to be multiplied by the molar mass constant $M_u = 0.99999999965 \pm 0.00000000030 g.mol^{-1}$ in order to get the value in $g.mol^{-1}$.

CHAPTER

THREE

RELEASE NOTES

3.1 0.1.0

3.1.1 Summary

- All elements from the periodic table added for the saw module.
- They are implemented as parameter derived type.

3.1.2 Download

iapws

pyiapws

3.1.3 Contributors

Milan Skocic

3.1.4 Commits

Full Changelog: https://github.com/MilanSkocic/ciaaw/compare/....0.1.0

CHAPTER

FOUR

API

4.1 ciaaw

Fortran and C API

4.2 pyciaaw

4.2.1 SAW (Standard atomic weights)

All elements as declared in the *ciaaw* are inserted at the top level of the module. C extension for saw.

BIBLIOGRAPHY

[1] Thomas Prohaska, Johanna Irrgeher, Jacqueline Benefield, John K. Böhlke, Lesley A. Chesson, Tyler B. Coplen, Tiping Ding, Philip J. H. Dunn, Manfred Gröning, Norman E. Holden, Harro A. J. Meijer, Heiko Moossen, Antonio Possolo, Yoshio Takahashi, Jochen Vogl, Thomas Walczyk, Jun Wang, Michael E. Wieser, Shigekazu Yoneda, Xiang-Kun Zhu, and Juris Meija. Standard atomic weights of the elements 2021 (iupac technical report). *Pure and Applied Chemistry*, 94(5):573–600, 2022. URL: https://doi.org/10.1515/pac-2019-0603, doi:doi:10.1515/pac-2019-0603.

[2] Adriaan M. H. van der Veen, Juris Meija, Antonio Possolo, and David Brynn Hibbert. Interpretation and use of standard atomic weights (iupac technical report). *Pure and Applied Chemistry*, 93(5):629–646, 2021. URL: https://doi.org/10.1515/pac-2017-1002, doi:doi:10.1515/pac-2017-1002.

PYTHON MODULE INDEX

INDEX

M

P