dxraylib

Release 0.0.1

Chris Ninham

Mar 29, 2023

CONTENTS:

CHAPTER

ONE

INTRODUCTION

dxraylib is a differentiable python reimplementation of the xraylib library for X-ray interactions with matter.

CHAPTER

TWO

INSTALLATION

TODO PyPi and conda.

CHAPTER

THREE

API

The api closely follows that of xraylib, the documentation for which can be found here.

3.1 Atomic weights

Relevant section in the xraylib documentation.

dxraylib.**AtomicWeight**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Standard atomic weight (g/mol).

Parameters
Z (*array_like*) – atomic number

Returns
standard atomic weight (g/mol)

Return type
array

3.2 Element densities

Relevant section in the xraylib documentation.

dxraylib.**ElementDensity**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Element density (g/cm^3) at room temperature.

Parameters
Z (*array_like*) – atomic number

Returns
element density (g/cm^3)

Return type
array

3.3 Cross-sections

Relevant section in the xraylib documentation.

dxraylib.**CS_Total**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Total cross-section (cm^2/g): Photoelectric + Compton + Rayleigh.

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)

Returns
Total cross-section (cm^2/g): Photoelectric + Compton + Rayleigh.

Return type
array

dxraylib.**CS_Photo**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Photoelectric absorption cross-section (cm^2/g).

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)

Returns
Photoelectric absorption cross-section (cm^2/g)

Return type
array

dxraylib.**CS_Rayl**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Rayleigh scattering cross-section (cm^2/g)

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)

Returns
Rayleigh scattering cross-section (cm^2/g)

Return type
array

dxraylib.**CS_Compt**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Compton scattering cross-section (cm^2/g)

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)

Returns
Compton scattering cross-section (cm^2/g)

Return type
array

dxraylib.**CSb_Total**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Total cross-section (barn/atom): Photoelectric + Compton + Rayleigh.

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)

Returns
Total cross-section (barn/atom): Photoelectric + Compton + Rayleigh

Return type
array

dxraylib.**CSb_Photo**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Photoelectric absorption cross-section (barn/atom).

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)

Returns
Photoelectric absorption cross-section (barn/atom)

Return type
array

dxraylib.**CSb_Rayl**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Rayleigh scattering cross-section (barn/atom).

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)

Returns
Rayleigh scattering cross-section (barn/atom)

Return type
array

dxraylib.**CSb_Compt**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Compton scattering cross-section (barn/atom).

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)

Returns
Compton scattering cross-section (barn/atom)

Return type
array

dxraylib.**CS_KN**(*E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Total Klein-Nishina cross-section (barn).

Parameters
E (*array_like*) – energy (keV)

Returns
total Klein-Nishina cross-section (barn)

Return type
array

dxraylib.**CS_Energy**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Mass-energy absorption cross-section (cm^2/g).

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)

Returns
Mass-energy absorption cross-section (cm^2/g)

Return type
array

dxraylib.**CS_Total_CP**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Total cross-section (cm^2/g): Photoelctric + Compton + Rayleigh.

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)

Returns
Total cross-section (cm^2/g): Photelectric + Compton + Rayleigh

Return type
array

dxraylib.**CS_Photo_CP**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Photoelectric absorption cross-section (cm^2/g).

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)

Returns
Photoelectric absorption cross-section (cm^2/g)

Return type
array

dxraylib.**CS_Rayl_CP**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Rayleigh scattering cross-section (cm^2/g).

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)

Returns
Rayleigh scattering cross-section (cm^2/g)

Return type
array

dxraylib.**CS_Compt_CP**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Compton scattering cross-section (cm^2/g).

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)

Returns
Compton scattering cross-section (cm^2/g)

Return type
array

dxraylib.**CSb_Total_CP**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Total cross-section (barn/atom) -> Photoelectric + Compton + Rayleigh.

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)

Returns
Total cross-section (barn/atom) -> Photoelectric + Compton + Rayleigh

Return type
array

dxraylib.**CSb_Photo_CP**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Photoelectric absorption cross-section (barn/atom).

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)

Returns
Photoelectric absorption cross-section (barn/atom)

Return type
array

dxraylib.**CSb_Rayl_CP**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Rayleigh scattering cross-section (barn/atom).

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)

Returns
Rayleigh scattering cross-section (barn/atom)

Return type
array

dxraylib.**CSb_Compt_CP**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Compton scattering cross-section (barn/atom).

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)

Returns
Compton scattering cross-section (barn/atom)

Return type
array

dxraylib.**CS_Energy_CP**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Mass-energy absorption cross-section (cm^2/g).

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)

Returns
Mass-energy absorption cross-section (cm^2/g)

Return type
array

3.4 Unpolarized differential scattering cross-sections

Relevant section in the xraylib documentation.

dxraylib.**DCS_Thoms**(*theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Thomson differential scattering cross-section (barn).

Parameters
theta (*array_like*) – scattering polar angle (rad)

Returns
Thomson differential scattering cross-section (barn)

Return type
array

dxraylib.**DCS_KN**(*E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Klein-Nishina differential scattering cross-section (barn).

Parameters

- **E** (*array_like*) – Energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)

Returns
Klein-Nishina differential scattering cross-section (barn)

Return type
array

dxraylib.**DCS_Rayl**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Rayleigh differential scattering cross-section ($cm^2/g/sr$).

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)

Returns
Rayleigh differential scattering cross-section ($cm^2/g/sr$)

Return type
array

dxraylib.**DCSb_Rayl**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Rayleigh differential scattering cross-section (barn/atom/sr).

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)

Returns
Rayleigh differential scattering cross-section (barn/atom/sr)

Return type
array

dxraylib.**DCS_Compt**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Compton differential scattering cross-section ($cm^2/g/sr$).

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)

Returns
Compton differential scattering cross-section ($cm^2/g/sr$)

Return type
array

dxraylib.**DCSb_Compt**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Compton differential scattering cross-section (barn/atom/sr).

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)

Returns
Compton differential scattering cross-section (barn/atom/sr)

Return type
array

dxraylib.**DCS_Rayl_CP**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Rayleigh differential scattering cross-section (cm^2/g/sr).

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)

Returns
Rayleigh differential scattering cross-section (cm^2/g/sr)

Return type
array

dxraylib.**DCS_Compt_CP**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Compton differential scattering cross-section (cm^2/g/sr).

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)

Returns
Compton differential scattering cross-section (cm^2/g/sr)

Return type
array

dxraylib.**DCSb_Rayl_CP**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Rayleigh differential scattering cross-section (barn/atom/sr).

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)

Returns
Rayleigh differential scattering cross-section (barn/atom/sr)

Return type
array

dxraylib.**DCSb_Compt_CP**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Compton differential scattering cross-section (barn/atom/sr).

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)

Returns
Compton differential scattering cross-section (barn/atom/sr)

Return type
array

3.5 Polarized differential scattering cross-sections

Relevant section in the xraylib documentation.

dxraylib.**DCSP_Thoms**(*theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *phi: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Thomson differential scattering cross-section for a polarized beam (barn).

Parameters

- **theta** (*array_like*) – scattering polar angle (rad)
- **phi** (*array_like*) – scattering azimuthal angle (rad)

Returns
Thomson differential scattering cross-section for a polarized beam (barn)

Return type
array

dxraylib.**DCSP_KN**(*E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *phi: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Klein-Nishina differential scattering cross-section for a polarized beam (barn).

Parameters

- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)
- **phi** (*array_like*) – scattering azimuthal angle (rad)

Returns
Klein-Nishina differential scattering cross-section for a polarized beam (barn)

Return type
array

dxraylib.**DCSP_Rayl**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *phi: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Rayleigh differential scattering cross-section for a polarized beam (cm^2/g/sr).

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)
- **phi** (*array*) – scattering azimuthal angle (rad)

Returns
Rayleigh differential scattering cross-section for a polarized beam (cm^2/g/sr)

Return type
array

dxraylib.**DCSP_Compt**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *phi: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Compton differential scattering cross-section for a polarized beam ($cm^2/g/sr$).

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)
- **phi** (*array_like*) – scattering azimuthal angle (rad)

Returns
Compton differential scattering cross-section for a polarized beam ($cm^2/g/sr$)

Return type
array

dxraylib.**DCSPb_Rayl**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *phi: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Rayleigh differential scattering cross-section for a polarized beam (barn/atom/sr).

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)
- **phi** (*array_like*) – scattering azimuthal angle (rad)

Returns
Rayleigh differential scattering cross-section for a polarized beam (barn/atom/sr)

Return type
array

dxraylib.**DCSPb_Compt**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *phi: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Compton differential scattering cross-section for a polarized beam (barn/atom/sr).

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)

- **phi** (*array_like*) – scattering azimuthal angle (rad)

Returns
Compton differential scattering cross-section for a polarized beam (barn/atom/sr)

Return type
array

dxraylib.**DCSP_Rayl_CP**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *phi: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Rayleigh differential scattering cross-section for a polarized beam (cm^2/g/sr).

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)
- **phi** (*array_like*) – scattering azimuthal angle (rad)

Returns
Rayleigh differential scattering cross-section for a polarized beam (cm^2/g/sr)

Return type
array

dxraylib.**DCSP_Compt_CP**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *phi: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Compton differential scattering cross-section for a polarized beam (cm^2/g/sr).

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)
- **phi** (*array_like*) – scattering azimuthal angle (rad)

Returns
Compton differential scattering cross-section for a polarized beam (cm^2/g/sr)

Return type
array

dxraylib.**DCSPb_Rayl_CP**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *phi: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Rayleigh differential scattering cross-section for a polarized beam (barn/atom/sr).

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)
- **phi** (*array_like*) – scattering azimuthal angle (rad)

Returns
Rayleigh differential scattering cross-section for a polarized beam (barn/atom/sr)

Return type
array

dxraylib.**DCSPb_Compt_CP**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *phi: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Compton differential scattering cross-section for a polarized beam (barn/atom/sr).

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)
- **phi** (*array_like*) – scattering azimuthal angle (rad)

Returns
Compton differential scattering cross-section for a polarized beam (barn/atom/sr)

Return type
array

3.6 Scattering factors

Relevant section in the xraylib documentation.

Included from the xraylib documentation: In this section, we introduce the momentum transfer parameter q, which is used in several of the following functions. It should be noted that several definitions can be found of this parameter throughout the scientific literature, which vary mostly depending on the community where it is used. The crystallography and diffraction community for example, use the following definition:

$$q = \frac{4\pi \times sin(\theta)}{\lambda}$$

with θ the angle between the incident X-ray and the crystal scattering planes according to Bragg's law, and λ the wavelength. xraylib (and dxraylib) uses however, a different definition, in which θ corresponds to the scattering angle ,

which in case of Bragg scattering is equal to twice the angle from the previous definition. This new definition has the advantage of being useful when working with amorphous materials, as well as with incoherent scattering. Furthermore, our definition drops the 4π scale factor, in line with the definition by Hubbell et al in Atomic form factors, incoherent scattering functions, and photon scattering cross-sections, J. Phys. Chem. Ref. Data, Vol.4, No. 3, 1975:

$$q = Ehc \times sin(\frac{\theta}{2}) \times 10^8$$

with E the energy of the photon, h Planck's constant and c the speed of light. The unit of the returned momentum transfer is then Å^{-1}.

dxraylib.**FF_Rayl**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *q: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Atomic form factor for Rayleigh scattering.

Parameters

- **Z** (*array_like*) – atomic number
- **q** (*array_like*) – momentum transfer (Å^{-1})

Returns
atomic form factor for Rayleigh scattering

Return type
array

dxraylib.**SF_Compt**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *q: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Incoherent scattering function for Compton scattering.

Parameters

- **Z** (*array_like*) – atomic number
- **q** (*array_like*) – momentum transfer (Å^{-1})

Returns
incoherent scattering function for Compton scattering

Return type
array

dxraylib.**MomentTransf**(*E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Momentum transfer for X-ray photon scattering (Å^{-1}).

Parameters

- **E** (*array_like*) – energy (keV)
- **theta** (*array_like*) – scattering polar angle (rad)

Returns
momentum transfer for X-ray photon scattering ($Å^{-1}$)

Return type
array

dxraylib.**Fi**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Anomalous scattering factor Δf'.

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)

Returns
anomalous scattering factor Δf'

Return type
array

dxraylib.**Fii**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Anomalous scattering factor Δf".

Parameters

- **Z** (*array_like*) – atomic number
- **E** (*array_like*) – energy (keV)

Returns
anomalous scattering factor Δf"

Return type
array

3.7 X-ray fluorescence line energies

Relevant section in the xraylib documentation.

3.7.1 Not yet implemented!

3.8 X-ray fluorescence yields

Relevant section in the xraylib documentation.

dxraylib.**FluorYield**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *shell: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Fluoresence yield.

Parameters

- **Z** (*array_like*) – atomic number
- **shell** (*array_like*) – shell-type macro

Returns
fluorescence yield

Return type
array

3.9 Auger yields

Relevant section in the xraylib documentation.

dxraylib.**AugerYield**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *shell: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Auger yield.

Parameters

- **Z** (*array_like*) – atomic number
- **shell** (*array_like*) – shell-type macro

Returns
auger yield

Return type
array

3.10 Coster-Kronig transition probabilities

Relevant section in the xraylib documentation.

dxraylib.**CosKronTransProb**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *trans: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Coster-Kronig transition probability.

Parameters

- **Z** (*array_like*) – atomic number
- **trans** (*array_like*) – Coster-Kronig transition macro

Returns
Coster-Kronig transition probability

Return type
array

3.11 Absorption edge energies

Relevant section in the xraylib documentation.

dxraylib.**EdgeEnergy**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *shell: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Absorption edge energy (keV).

Parameters

- **Z** (*array_like*) – atomic number
- **shell** (*array_like*) – shell-type macro

Returns
absorption edge energy (keV)

Return type
array

3.12 Jump factors

Relevant section in the xraylib documentation.

dxraylib.**JumpFactor**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *shell: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Jump factor.

Parameters

- **Z** (*array_like*) – atomic number
- **shell** (*array_like*) – shell-type macro

Returns
jump factor

Return type
array

3.13 X-ray fluorescence cross-sections

Relevant section in the xraylib documentation.

3.13.1 Not yet implemented!

3.14 Radiative rates

Relevant section in the xraylib documentation.

dxraylib.**RadRate**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *line: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Radiative rate.

Parameters

- **Z** (*array_like*) – atomic number
- **line** (*array_like*) – line-type macro

Returns
radiative rate

Return type
array

3.15 Non-radiative rates

Relevant section in the xraylib documentation.

dxraylib.**AugerRate**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *auger_trans: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Non-radiative rate.

Parameters

- **Z** (*array_like*) – atomic number
- **auger_trans** (*array_like*) – Auger-type macro corresponding with the electrons involved

Returns
non-radiative rate

Return type
array

3.16 Atomic level widths

Relevant section in the xraylib documentation.

dxraylib.**AtomicLevelWidth**(*Z: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *shell: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Atomic level width (keV).

Parameters

- **Z** (*array_like*) – atomic number
- **shell** (*array_like*) – shell-type macro

Returns
atomic level width (keV)

Return type
array

3.17 Compton energy

Relevant section in the xraylib documentation.

dxraylib.**ComptonEnergy**(*E0: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *theta: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Photon energy after Compton scattering (keV).

Parameters

- **E0** (*array_like*) – photon energy before scattering (keV)
- **theta** (*array_like*) – scattering polar angle (rad)

Returns
photon energy after Compton scattering (keV)

Return type
array

3.18 Refractive indices

Relevant section in the xraylib documentation.

dxraylib.**Refractive_Index_Re**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *density: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Real component of the refractive index: 1 - δ.

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)
- **density** (*array_like*) – density (g/cm^3)

Returns
real component of the refractive index: 1 - δ

Return type
array

dxraylib.**Refractive_Index_Im**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *density: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Imaginary component of the refractive index: β.

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)
- **density** (*array_like*) – density (g/cm^3)

Returns
imaginary component of the refractive index: β

Return type
array

dxraylib.**Refractive_Index**(*compound: str*, *E: Union[Array, ndarray, bool_, number, bool, int, float, complex]*, *density: Union[Array, ndarray, bool_, number, bool, int, float, complex]*) → Array

Complex refractive index: 1 - δ + iβ.

Parameters

- **compound** (*str*) – chemical formula or NIST compound name
- **E** (*array_like*) – energy (keV)
- **density** (*array_like*) – density (g/cm^3)

Returns
complex refractive index: 1 - δ + iβ

Return type
array

3.19 Compton profiles

Relevant section in the xraylib documentation.

3.19.1 Not yet implemented!

3.20 Electronic configurations

Relevant section in the xraylib documentation.

3.20.1 Not yet implemented!

3.21 Crystal diffraction

Relevant section in the xraylib documentation.

3.21.1 Not yet implemented!

3.22 Compound parser

Relevant section in the xraylib documentation.

class dxraylib.xraylib_parser.**compoundData**

Bases: TypedDict

A compound dataset.

Elements: tuple[int, ...]

a tuple (length = nElements) containing the elements in ascending order

massFractions: tuple[float, ...]

a tuple (length = nElements) containg the mass fractions of the elements in Elements

molarMass: float

the molar mass of the compound, in g/mol

nAtoms: tuple[float, ...]

a tuple (length = nElements) containing the number of atoms each element has in the compound

nAtomsAll: float

number of atoms in the formula. Since indices may be real numbers, this attribute is of type float

nElements: int

number of different elements in the compound

3.22.1 CompoundParser

The CompoundParser function will parse a chemical formula compoundString and will allocate a compoundData structure with the results if successful, otherwise NULL is returned. Chemical formulas may contain (nested) brackets, followed by an integer or real number (with a dot) subscript. Examples of accepted formulas are: H2O, Ca5(PO4)3F, Ca5(PO4)F0.33Cl0.33(OH)0.33.

dxraylib.**CompoundParser**(*compoundString: str*) → *compoundData*

Wrapper around xraylib.CompoundParser

Parse a chemical formula compoundString to a compoundData dictionary. Chemical formulas may contain (nested) brackets, followed by an integer or real number (with a dot) subscript. Examples of accepted formulas are: H20, Ca5(PO4)3F, Ca5(PO4)F0.33Cl0.33(OH)0.33

Parameters
compoundString (*str*) – chemical formula to parse

Returns
description

Return type
compoundData

Raises
ValueError – If compoundString is an invalid chemical formula

3.22.2 AtomicNumberToSymbol

The AtomicNumberToSymbol function returns a pointer to a string containing the element for atomic number Z. If an error occurred, the NULL string is returned.

dxraylib.**AtomicNumberToSymbol**(*Z: int*) → str

Wrapper around xraylib.AtomicNumberToSymbol

Return the symbol for the element of atomic number Z.

Parameters
Z (*int*) – atomic number

Returns
element symbol

Return type
str

Raises
ValueError – if Z is an invalid atomic number: Z<1 ^ Z>107

3.22.3 SymbolToAtomicNumber

The SymbolToAtomicNumber function returns the atomic number that corresponds with element symbol. If the element does not exist, 0 is returned.

dxraylib.**SymbolToAtomicNumber**(*symbol: str*) → int

Wrapper around xraylib.SymbolToAtomicNumber

Return the atomic number for the elemtent with symbol: symbol.

Parameters
symbol (*str*) – element symbol

Returns
atomic number

Return type
int

Raises
ValueError – if symbol is an invalid element symbol or corresponds to an element with Z>107

3.23 NIST compound catalogue

Relevant section in the xraylib documentation.

class dxraylib.xraylib_nist_compounds.**compoundDataNIST**

Bases: TypedDict

A NIST compound dataset.

Elements: tuple[int, ...]
a tuple (length = nElements) containing the elements, in ascending order

density: float
the density of the compound, expressed in g/cm^3

massFractions: tuple[float, ...]
a tuple (length = nElements) containg the mass fractions of the elements in Elements

nElements: int
number of different elements in the compound

name: str
a string containing the full name of the compound, as retrieved from the NIST database

3.23.1 GetCompoundDataNISTByName

dxraylib.**GetCompoundDataNISTByName**(*compoundString: str*) → *compoundDataNIST*

summary

Parameters
compoundString (*str*) – _description_

Returns
description

Return type
compoundDataNIST

3.23.2 GetCompoundDataNISTByIndex

dxraylib.**GetCompoundDataNISTByIndex**(*compoundIndex: int*) → *compoundDataNIST*

summary

Parameters
compoundIndex (*int*) – _description_

Returns
description

Return type
compoundDataNIST

3.23.3 GetCompoundDataNISTList

dxraylib.**GetCompoundDataNISTList**() → tuple[str, ...]

summary

Returns
description

Return type
tuple[str, . . .]

3.24 Radionuclides

Relevant section in the xraylib documentation.

class dxraylib.xraylib_radionuclides.**radioNuclideData**

Bases: TypedDict

A radionuclide dataset.

A: int
mass number of the radionuclide

GammaEnergies: tuple[float, ...]
a tuple (length = nGammas) of emitted gamma-ray energies

GammaIntensities: tuple[float, ...]
a tuple (length = nGammas) of emitted gamma-ray photons per disintegration

N: int
number of neutrons of the radionuclide

XrayIntensities: tuple[float, ...]
a tuple (length = nXrays) of photons per disintegration, one value per emitted X-ray

XrayLines: tuple[int, ...]
a tuple (length = nXrays) of line-type macros, identifying the emitted X-rays

Z: int
atomic number of the radionuclide

Z_xray: int

atomic number of the nuclide after decay, which should be used in calculating the energy of the emitted X-ray lines using LineEnergy

nGammas: int

number of emitted gamma-rays

nXrays: int

number of emitted characteristic X-rays

name: str

a string containing the mass number (A), followed by the chemical element (e.g. 55Fe)

3.24.1 GetRadioNuclideDataByName

dxraylib.**GetRadioNuclideDataByName**(*radioNuclideString: str*) → *radioNuclideData*

summary

Parameters
radioNuclideString (*str*) – _description_

Returns
description

Return type
radioNuclideData

3.24.2 GetRadioNuclideDataByIndex

dxraylib.**GetRadioNuclideDataByIndex**(*radioNuclideIndex: int*) → *radioNuclideData*

summary

Parameters
radioNuclideIndex (*int*) – _description_

Returns
description

Return type
radioNuclideData

3.24.3 GetRadioNuclideDataList

dxraylib.**GetRadioNuclideDataList**() → tuple[str, ...]

summary

Returns
description

Return type
tuple[str, . . .]

3.25 Constants

Relevant section in the xraylib documentation.

TODO

INDEX