# atomicdataMB

*Release 1.1.9*

**Matthew Burger**

**Sep 24, 2020**

# CONTENTS

atomicdataMB provides support for the Neutral Cloud and Exospheres Model (nexoclom) and can be used as a standalone package.

CHAPTER

# ONE

# ATOMICMASS

`atomicmass` - Return the atomic mass of an atom or molecule.

This is really just a wrapper for periodictable (https://periodictable.readthedocs.io/en/latest/index.html) but returns the mass as an astropy quantity (http://docs.astropy.org/en/stable/units/index.html).

`atomicdataMB.atomicmass.`**`atomicmass`**(*species*)

Return the atomic mass of an atom or molecule.

**Parameters**

**species** Chemical formula requested species. See periodictable (https://periodictable.readthedocs.io/en/latest/index.html) for formatting options.

**Returns**

The atomicmass of *species* as an astropy quantity with units = AMU ($1\,\mathrm{AMU} = 1.660539 \times 10^{-27}\,\mathrm{kg}$). If `periodictable` returns a ValueError, *None* is returned.

**Examples**

```
>>> from atomicdataMB import atomicmass
>>> print(atomicmass('Na'))
22.98977 u
>>> print(atomicmass('H2O'))
18.01528 u
>>> print(atomicmass('X'))
WARNING: mathMB.atomicmass: X not found
None
```

CHAPTER

TWO

# G_VALUES

`g_values` - Routines related to g-values and radiation pressure The g-value is the the product of the solar flux at the dopler-shifted emission wavelength and the scattering probability per atom. See Killen, R.M. et al., Icarus 209, 75–87, 2009. (http://dx.doi.org/10.1016/j.icarus.2010.02.018.) for details on calculating g-values for important species in Mercury's atmosphere.

The radiation acceleration is given by $a_{rad} = hg/m\lambda$, where h is Plank's constant, g is the g-value as a function of radial velocity, m is the mass of the accelerated species, and  is the wavelength of the absorbed photon.

**class** `atomicdataMB.g_values.`**RadPresConst**(*sp*, *aplanet*)

Class containing radial acceleration vs. velocity for a specified atom.

**Parameters**

**sp** atomic species

**aplanet** Distance from the Sun. Can be given as an astropy quantity with distance units or as a float assumed to be in AU. Default = 1 AU

**database** Database containing solar system information. Default = `thesolarsystem` which probably shouldn't be overridden.

**Class Attributes**

**species** The input species

**aplanet** The input distance from the Sun

**velocity** Radial velocity deviation relative to the Sun in km/s. Positive values indicate motion away from the Sun. Given as a numpy array of astropy quantities

**accel** Radial acceleration vs. velocity with units km/s**2.

**class** `atomicdataMB.g_values.`**gValue**(*sp*, *wavelength=None*, *aplanet=<Quantity 1. AU>*)

Class containing g-value vs. velocity for a specified atom and transition.

**Parameters**

**sp** atomic species

**wavelength** Wavelength of the transition. Default=None.

**aplanet** Distance from the Sun. Can be given as an astropy quantity with distance units or as a float assumed to be in AU. Default = 1 AU

**Class Attributes**

**species** The input species

**wavelength** The input wavelength

**aplanet** The input aplanet

**velocity** Radial velocity deviation relative to the Sun in km/s. Positive values indicate motion away from the Sun. Given as a numpy array of astropy quantities

**g** g-value as function of velocity in units 1/s.

atomicdataMB.g_values.**make_gvalue_table**()

Creates and populates gvalues database table. Creates a database table called gvalues. Fields in the table:

**filename** Filename in project tree containing the data; used only for populating the database

**reference** Source of the g-values

**species** Atomic species

**refpt** Distance from Sun in AU for which the g-values were calculated.

**wavelength** At-rest wavelength for the g-values in Å.

**velocity** $\Delta$v from rest in km/s.

**g** g-values in photons/s as a fucntion of velocity.

**Parameters**

None

**Returns**

No Output

CHAPTER

# THREE

# PHOTOLOSSRATES

`photolossrates` - Determine photoionization and photodissociation rates

**class** `atomicdataMB.photolossrates.`**PhotoRate**(*species*, *aplanet_=<Quantity 1. AU>*)

Determine photoreactions and photorates for a species.

**Parameters**

**species** Species to compute rates for.

**aplanet** Distance from the Sun. Default is 1 AU. Given as either a numeric type or an astropy quantity with length units.

**Class Attributes**

**species** Species

**aplanet** Distance from the Sun; astropy quantity with units AU

**rate** Reaction rate; astropy quantity with units s^{-1}. Rate is the sum of all possible reactions for the species.

**reactions** Pandas dataframe with columns for reaction and rate (in s^{-1}) for each reaction for the species. This can be used to determine the products produced by photolysis and photoionization.

**Example**

```
>>> from atomicdataMB import PhotoRate
>>> kappa = PhotoRate('Na', 0.33)
>>> print(kappa)
Species = Na
Distance = 0.33 AU
Rate = 6.666666666666666e-05 1 / s
>>> print(kappa.rate)
6.666666666666666e-05 1 / s
>>> print(kappa.reactions)
              reaction                  kappa
0  Na, photon -> Na+, e  6.666666666666666e-05
>>> kappa = PhotoRate('H_2O')
>>> print(kappa)
Species = H_2O
Distance = 1.0 AU
Rate = 1.2056349999999999e-05 1 / s
>>> print(kappa.reactions)
                    reaction     kappa
0      H_2O, photon -> H_2, O  5.97e-07
1       H_2O, photon -> OH, H  1.03e-05
2     H_2O, photon -> H, H, O  7.54e-07
3   H_2O, photon -> H, OH+, e  5.54e-08
```

(continues on next page)

(continued from previous page)

```
4   H_2O, photon -> OH, H+, e  1.31e-08
5    H_2O, photon -> H_2O+, e  3.31e-07
6  H_2O, photon -> H_2, O+, e  5.85e-09
```

atomicdataMB.photolossrates.**make_photo_table**()

Creates and populates photorates database table. Creates a database table called photorates. Fields in the table:

**filename** Filename in project tree containing the data; used only for populating the database

**reference** Source of the photoionization or photodissociation rate

**species** Atomic or molecular species

**reaction** Photoionization or photodissociation reaction

If multiple reaction rates are found for a reaction, user is prompted to choose the best one. Most of the reactions are in: Huebner & Mukherjee (2015), Astrophys. Space Sci., 195, 1-294.

**Parameters**

None

**Returns**

No output

CHAPTER

FOUR

# INITIALIZE_ATOMICDATA

Populate the database with the available atomic data. Currently populates g-values and photoionization rates. If the database does not exist, it will be created. By default, the tables will only be created if

atomicdataMB.initialize_atomicdata.**initialize_atomicdata**(*force=False*)

Populate the database with available atomic data if nececssary.

**Parameters**

**force** By default, the database tables are only created if they do not already exist. Set force to True to force the tables to be remade. This would be necessary if there are updates to the atomic data.

**Output**

No output.

CHAPTER

FIVE

# DATABASE_CONNECT

`database_connect` - Return a database connection to saved atomic data

`atomicdataMB.database_connect.`**`database_connect`**(*database=None*, *port=None*, *return_con=True*)

Return a database connection to saved atomic data Wrapper for `psycopg2.connect()` that determines database and port to use.

**Parameters**

**database** Database to connect to. If not given, it must be supplied in the $HOME/.nexoclom configuration file.

**port** Port the database server uses. If not given, it must be supplied in the $HOME/.nexoclom configuration file.

**return_con** False to return database name and port instead of connection. Default = True

**Returns**

Database connection with autocommit = True unless return_con = False

**Examples**

```
>>> from atomicdataMB import database_connect
>>> database, port = database_connect(return_con=False)
>>> print(f'database = {database}; port = {port}')
database = thesolarsystemmb; port = 5432
>>> with database_connect() as con:
...     cur = con.cursor()
...     cur.execute('SELECT DISTINCT species from gvalues')
...     species = cur.fetchall()
>>> species = [s[0] for s in species]
>>> print(species)
['Ca', 'OH', 'O', 'Ti', 'C', 'Mg+', 'Na', 'Mg', 'H', 'Mn', 'He',
 'Ca+', 'K', 'S']
```

CHAPTER

SIX

# INSTALLATION

atomicdataMB can be installed with pip:

```
$ pip install atomicdataMB
```

CHAPTER

SEVEN

# REPORTING ISSUES

This project is hosted on github at atomicdataMB (https://github.com/mburger-stsci/atomicdataMB). Please report bugs or make comments there.

CHAPTER

EIGHT

# CONTRIBUTING

Please let me know if you would like to make contributions.

**Authors** Matthew Burger

**License** LICENSE

# PYTHON MODULE INDEX

## a

# INDEX