AACGM-v2 Python library

Release 2.6.0

Aug 16, 2020

Contents

CHAPTER 1

Overview

This is a Python wrapper for the AACGM-v2 C library, which allows converting between geographic and magnetic coordinates. The currently included version of the C library is 2.6. The wrapper is provided "as is" in the hopes that it will be useful to the space science community, and will not automatically be updated when new versions of the C library is released. MLT calculations are included in the wrapper (not part of the C library, please see the documentation for implementation details). The package is free software (MIT license).

This fork of aacgmv2 (github link) provides the interface compatible with `aacgmv2 < 2.4`.

1.1 Quick start

Install (requires NumPy):

```
pip install aacgm2
```

Convert between AACGM and geographic coordinates:

```
>>> from aacgm2 import convert
>>> from datetime import date
>>> # geo to AACGM, single numbers
>>> mlat, mlon, malt = convert(60, 15, 300, date(2013, 11, 3))
>>> "{0:.8f}".format(float(mlat))
'57.47357891'
>>> "{0:.8f}".format(float(mlon))
'93.61113360'
>>> "{0:.8f}".format(float(malt))
'1.04566346'
>>> # AACGM to geo, mix arrays/numbers
>>> glat, glon, galt = convert([90, -90], 0, 0, date(2013, 11, 3), a2g=True)
>>> ["{0:.8f}".format(float(gl)) for gl in glat]
['82.96859922', '-74.33899667']
```

(continues on next page)

(continued from previous page)

```
>>> ["{0:.8f}".format(float(gl)) for gl in glon]
['-84.65010944', '125.84759847']
>>> ["{0:.8f}".format(float(ga)) for ga in galt]
['14.12457922', '12.87721946']
```

Convert between AACGM and MLT:

```
>>> from aacgm2 import convert_mlt
>>> from datetime import datetime
>>> mlon = convert_mlt([0, 12], datetime(2013, 11, 3, 18, 0), m2a=True)
>>> ["{0:.8f}".format(float(ml)) for ml in mlon]
['159.08043649', '339.08043649']
```

If you don't know or use Python, you can also use the command line. See details in the full documentation.

1.2 Documentation

https://aacgm2.readthedocs.org/

1.3 Badges

docs	
tests	
package	

CHAPTER 2

Installation

This package requires NumPy, which you can install alone or as a part of SciPy. Some Python distributions come with NumPy/SciPy pre-installed. For Python distributions without NumPy/SciPy, Windows/Mac users should install pre-compiled binaries of NumPy/SciPy, and Linux users may have NumPy/SciPy available in their repositories.

When you have NumPy, install this package at the command line using `pip`[1]:

```
pip install aacgm2
```

The package has been tested with the following setups (others might work, too):

- Windows (32/64 bit) and Linux (64 bit)
- Python 2.7, 3.3, 3.4 and 3.5
- NumPy 1.8, 1.9, 1.10

[1] pip is included with Python 2 from v2.7.9 and Python 3 from v3.4. If you don't have pip, get it here.

CHAPTER 3

Usage examples

3.1 Python library

For full documentation of the functions, see *Reference → aacgm2*.

```
>>> from aacgm2 import convert
>>> from datetime import date
>>> # geo to AACGM, single numbers
>>> mlat, mlon, malt = convert(60, 15, 300, date(2013, 11, 3))
>>> "{0:.8f}".format(float(mlat))
'57.47357891'
>>> "{0:.8f}".format(float(mlon))
'93.61113360'
>>> "{0:.8f}".format(float(malt))
'1.04566346'
>>> # AACGM to geo, mix arrays/numbers
>>> glat, glon, galt = convert([90, -90], 0, 0, date(2013, 11, 3), a2g=True)
>>> ["{0:.8f}".format(float(gl)) for gl in glat]
['82.96859922', '-74.33899667']
>>> ["{0:.8f}".format(float(gl)) for gl in glon]
['-84.65010944', '125.84759847']
>>> ["{0:.8f}".format(float(ga)) for ga in galt]
['14.12457922', '12.87721946']
```

3.2 Command-line interface

The Python package also installs a command called `aacgm2` with two sub-commands, `aacgm2 convert` and `aacgm2 convert_mlt`. The command-line interface allows you to make use of the Python library even if you don't know or use Python. See *Reference → Command-line interface* for a list of arguments to the commands. Below are some simple usage examples.

3.2.1 Convert geographical/magnetic coordinates

Produce a file called e.g. `input.txt` with the input latitudes, longitudes and altitudes on each row separated by whitespace:

```
# lat lon alt
# comment lines like these are ignored
60 15 300
61 15 300
62 15 300
```

To convert this to AACGM-v2 for the date 2015-02-24, run the command `aacgm2 convert -i input.txt -o output.txt -d 20150224`. The output file will look like this:

```
57.47612194 93.55719875
58.53323704 93.96069212
59.58522105 94.38968625
```

Alternatively, you can skip the files and just use command-line piping:

```
$ echo 60 15 300 | aacgm2 convert -d 20150224
57.47612194 93.55719875
```

3.2.2 Convert MLT

This works in much the same way as `convert`. The file should only contain a single column of numbers (MLTs or magnetic longitudes, depending on which way you're converting):

```
1
12
23
```

To convert these MLTs to magnetic longitudes at 2015-02-24 14:00:15, run e.g. `aacgm2 convert_mlt 20150224140015 -i input.txt -o output.txt -v` (note that the date/time is a required parameter). The output file will then look like this:

```
240.13651777
45.13651777
210.13651777
```

Like with `convert`, you can use stdin/stdout instead of input/output files:

```
$ echo 12 | aacgm2 convert_mlt 20150224140015 -v
45.13651777
```

CHAPTER 4

Reference

4.1 aacgm2

These functions are available when you `import aacgm2`.

`aacgm2.`**`convert`**(*lat*, *lon*, *alt*, *date=None*, *a2g=False*, *trace=False*, *allowtrace=False*, *badidea=False*, *geocentric=False*)

Converts to/from geomagnetic coordinates.

This is a user-friendly pythonic wrapper for the low-level C interface functions available in *aacgm2._aacgmv2*.

Parameters

- **`lat,lon,alt`** (*array_like*) – Input latitude(s), longitude(s) and altitude(s). They must be broadcastable to the same shape.
- **`date`** (`datetime.date`/`datetime.datetime`, optional) – The date/time to use for the magnetic field model, default `None` (uses current time). Must be between 1900 and 2020.
- **`a2g`** (*bool, optional*) – Convert from AACGM-v2 to geographic coordinates, default `False` (converts geographic to AACGM-v2).
- **`trace`** (*bool, optional*) – Use field-line tracing, default `False` (uses coefficients). Tracing is more precise and needed at altitudes > 2000 km, but significantly slower.
- **`allowtrace`** (*bool, optional*) – Automatically use field-line tracing above 2000 km, default `False` (raises an exception for these altitudes unless `trace=True` or `badidea=True`).
- **`badidea`** (*bool, optional*) – Allow use of coefficients above 2000 km (bad idea!)
- **`geocentric`** (*bool, optional*) – Assume inputs are geocentric with Earth radius 6371.2 km.

Returns

- **lat_out** (numpy.ndarray) – Converted latitude
- **lon_out** (numpy.ndarray) – Converted longitude
- **alt_out** (numpy.ndarray) – Converted altitude

Raises

- **ValueError** – if max(alt) > 2000 and neither of *trace*, *allowtrace*, or *badidea* is True
- **ValueError** – if latitude is outside the range -90 to +90 degrees
- **RuntimeError** – if there was a problem in the C extension

Notes

This function exclusively relies on the AACGM-v2 C library. Specifically, it calls the functions *_aacgmv2.setDateTime()* and *_aacgmv2.aacgmConvert()*, which are simple interfaces to the C library functions AACGM_v2_SetDateTime() and AACGM_v2_Convert(). Details of the techniques used to derive the AACGM-v2 coefficients are described by Shepherd, 2014 [1].

aacgm2.**convert_mlt**(*arr*, *datetime*, *m2a=False*)

Converts between magnetic local time (MLT) and AACGM-v2 longitude.

Note: This function is not related to the AACGM-v2 C library, but is provided as a convenience in the hopes that it might be useful for some purposes.

Parameters

- **arr** (*array_like or float*) – Magnetic longitudes or MLTs to convert.
- **datetime** (datetime.datetime) – Date and time for MLT conversion in Universal Time (UT).
- **m2a** (*bool*) – Convert MLT to AACGM-v2 longitude (default is False, which implies conversion from AACGM-v2 longitude to MLT).

Returns **out** – Converted coordinates/MLT

Return type numpy.ndarray

Notes

The MLT conversion is not part of the AACGM-v2 C library and is instead based on Laundal et al., 2016 [1]. A brief summary of the method is provided below.

MLT is defined as

MLT = (magnetic longitude - magnetic noon meridian longitude) / 15 + 12

where the magnetic noon meridian longitude is the centered dipole longitude of the subsolar point.

There are two important reasons for using centered dipole instead of AACGM for this calculation. One reason is that the AACGM longitude of the subsolar point is often undefined (being at low latitudes). More importantly, if the subsolar point close to ground was used, the MLT at polar latitudes would be affected by non-dipole features at low latitudes, such as the South Atlantic Anomaly. This is not desirable; since the Sun-Earth interaction takes place at polar field lines, it is these field lines the MLT should describe.

In calculating the centered dipole longitude of the subsolar point, we use the first three IGRF Gauss coefficients, using linear interpolation between the model updates every five years.

Both input and output MLON are taken modulo 360 to ensure they are between 0 and 360 degrees. Similarly, input/output MLT are taken modulo 24. For implementation of the subsolar point calculation, see *subsol()*.

aacgm2.**subsol**(*year*, *doy*, *ut*)

Finds subsolar geocentric longitude and latitude.

Helper function for *convert_mlt()*.

Parameters

- **year** (*int [1601, 2100]*) – Calendar year
- **doy** (*int [1, 365/366]*) – Day of year
- **ut** (*float*) – Seconds since midnight on the specified day

Returns

- **sbsllon** (*float*) – Subsolar longitude for the given date/time
- **sbsllat** (*float*) – Subsolar latitude for the given date/time

Notes

Based on formulas in Astronomical Almanac for the year 1996, p. C24. (U.S. Government Printing Office, 1994). Usable for years 1601-2100, inclusive. According to the Almanac, results are good to at least 0.01 degree latitude and 0.025 degrees longitude between years 1950 and 2050. Accuracy for other years has not been tested. Every day is assumed to have exactly 86400 seconds; thus leap seconds that sometimes occur on December 31 are ignored (their effect is below the accuracy threshold of the algorithm).

After Fortran code by A. D. Richmond, NCAR. Translated from IDL by K. Laundal.

aacgm2.**set_coeff_path**()

Sets the environment variables `AACGM_v2_DAT_PREFIX` and `IGRF_12_COEFFS` (for the current process). These are required for the C library to function correctly. This function is automatically called when importing aacgm2. You may need to call this manually if you use multithreading or spawn child processes (untested).

4.2 aacgm2._aacgmv2

This submodule contains the interface to the AACGM-v2 C library. For the user-friendly wrapper, see *aacgm2.convert()*.

This module contains the interface to the AACGM-v2 C library.

aacgm2._aacgmv2.**aacgmConvert**(*in_lat*, *in_lon*, *height*, *code*) → out_lat, out_lon, r

Converts between geographic and magnetic coordinates.

Parameters

- **in_lat** (*float [-90, 90]*) – Input latitude
- **in_lon** (*float [-180, 180]*) – Input longitude
- **height** (*float*) – Input altitude
- **code** (*int*) – Bitwise code for passing options into converter. The codes and their names (defined in this module) are given in the table below.

Returns

- **out_lat** (*float*) – Converted latitude

- **out_lon** (*float*) – Converted longitude
- **r** (*float*) – Not used, always 1.0

Notes

The bitwise codes are:

Code	Name	Description
0	G2A	Convert geographic to AACGM-v2.
1	A2G	Convert AACGM-v2 to geographic.
2	TRACE	Use field-line tracing instead of coefficients. More precise, but significantly slower.
4	ALLOW-TRACE	Automatically use field-line tracing above 2000 km. If not set, cause exception to be thrown for these altitudes unless TRACE or BADIDEA is set.
8	BA-DIDEA	Allow use of coefficients above 2000 km (bad idea!)
16	GEO-CEN-TRIC	Assume inputs are geocentric with Earth radius 6371.2 km.

For example, to convert from AACGM-v2 to geographpic using field-line tracing, use either of the following:

```
>>> aacgmConvert(in_lat, in_lon, height, A2G | TRACE)
>>> aacgmConvert(in_lat, in_lon, height, 1 | 2)
>>> aacgmConvert(in_lat, in_lon, height, 3)
```

aacgm2._aacgmv2.**setDateTime**(*year*, *month*, *day*, *hour*, *minute*, *second*)

Sets the date and time for the IGRF magnetic field.

Parameters

- **year** (*int [1900, 2020]*) –
- **month** (*int [1, 12]*) –
- **day** (*int [1, 31]*) –
- **hour** (*int [0, 24]*) –
- **minute** (*int [0, 60]*) –
- **second** (*int [0, 60]*) –

4.3 Command-line interface

When you install this package you will get a command called `aacgm2`. It has two subcommands, `convert` and `convert_mlt`, which correspond to the functions *aacgm2.convert()* and *aacgm2.convert_mlt()*. See the documentation for these functions for a more thorough explanation of arguments and behaviour.

You can get help on the two commands by running `aacgm2 convert -h` and `aacgm2 convert_mlt -h`.

4.3.1 convert

```
$ aacgm2 convert -h
usage: aacgm2 convert [-h] [-i FILE_IN] [-o FILE_OUT] [-d YYYYMMDD] [-v] [-t]
                      [-a] [-b] [-g]

optional arguments:
  -h, --help            show this help message and exit
  -i FILE_IN, --input FILE_IN
                        input file (stdin if none specified)
  -o FILE_OUT, --output FILE_OUT
                        output file (stdout if none specified)
  -d YYYYMMDD, --date YYYYMMDD
                        date for magnetic field model (1900-2020, default:
                        today)
  -v, --a2g             invert - convert AACGM to geographic instead of
                        geographic to AACGM
  -t, --trace           use field-line tracing instead of coefficients
  -a, --allowtrace      automatically use field-line tracing above 2000 km
  -b, --badidea         allow use of coefficients above 2000 km (bad idea!)
  -g, --geocentric      assume inputs are geocentric with Earth radius 6371.2
                        km
```

4.3.2 convert_mlt

```
$ aacgm2 convert_mlt -h
usage: aacgm2 convert_mlt [-h] [-i FILE_IN] [-o FILE_OUT] [-v] YYYYMMDDHHMMSS

positional arguments:
  YYYYMMDDHHMMSS        date and time for conversion

optional arguments:
  -h, --help            show this help message and exit
  -i FILE_IN, --input FILE_IN
                        input file (stdin if none specified)
  -o FILE_OUT, --output FILE_OUT
                        output file (stdout if none specified)
  -v, --m2a             invert - convert MLT to AACGM longitude instead of
                        AACGM longitude to MLT
```

CHAPTER 5

Contributing

Bug reports, feature suggestions and other contributions are greatly appreciated! While I can't promise to implement everything, I will always try to respond in a timely manner.

5.1 Short version

- Submit bug reports and feature requests at GitHub
- Make pull requests to the `develop` branch

5.2 Bug reports

When reporting a bug please include:

- Your operating system name and version
- Any details about your local setup that might be helpful in troubleshooting
- Detailed steps to reproduce the bug

5.3 Feature requests and feedback

The best way to send feedback is to file an issue at GitHub.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

5.4 Development

To set up *aacgm2* for local development:

1. Fork aacgm2 on GitHub.
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/aacgmv2.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally. Add tests for bugs and new features in `tests/test_py_aacgmv2.py` (for the wrapper), `test_c_aacgmv2.py` (for the C extension), or `tests/test_cmd_aacgmv2.py` (for the command-line interface). The tests are run with `py.test` and can be written as normal functions (starting with `test_`) containing a standard `assert` statement for testing output.

4. When you're done making changes, run all the checks, doc builder and spell checker with tox[1]:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Brief description of your changes"
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website. Pull requests should be made to the `develop` branch.

5.4.1 Pull Request Guidelines

One simple guideline: **Don't break userspace.** That is, changing the public interface in a backwards-incompatible way is **strongly discouraged**. Although this package is not as essential as an operating system kernel, remember that there may be people out there that use this package. Changing the interface in an incompatible way will break their setup and cause headaches. So be kind to others and avoid such cases as much as possible, it is exactly the reason why this fork exists at all. You can read more about why keeping an API stable is a good thing, and the kernel documentation.

If you need some code review or feedback while you're developing the code, just make a pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1]
2. Update/add documentation if relevant
3. Add a note to `CHANGELOG.rst` about the changes
4. Add yourself to `AUTHORS.rst`

[1] If you don't have all the necessary Python versions available locally or have trouble building NumPy in all the testing environments, you can rely on Travis and AppVeyor - they will run the tests for each change you add in the pull request.

5.4.2 Tips

To run a subset of tests:

```
tox -e envname -- py.test -k test_myfeature
```

To run all the test environments in parallel (you need to `pip install detox`):

```
detox
```

CHAPTER 6

Authors

- Stefan Bender - https://github.com/st-bender
- Angeline G. Burrell - https://github.com/aburrell
- Christer van der Meeren - https://github.com/cmeeren
- Karl M. Laundal

CHAPTER 7

Changelog

7.1 2.6.0 (2020-08-16)

- Same as 2.6.0-rc2

7.2 2.6.0-rc2 (2020-08-14)

- Binary wheels for MacOSX

7.3 2.6.0-rc1 (2020-08-12)

- Updated AACGM-v2 coefficients derived using the IGRF13 model
- Updated IGRF and GUFM1 coefficients using the IGRF13 model
- Added additional checks to the C code for reading the IGRF13 coefficient file
- Updated CI setup on travis and appveyor
- Deployment of linux and osx wheels to the package index
- Changed version support to 2.7, 3.4, 3.5, 3.6, 3.7, and 3.8
- Updated test values to match new coefficients

7.4 2.5.0 (unreleased)

- Updated C code and coefficients to version 2.5.

7.5 2.3.9 (2018-05-27)

- Update to AACGM-v2.4, which includes changes to the inverse MLT and dipole tilt functions and some minor bug fixes
- Updated dependencies
- Removed support for python 3.3

7.6 2.0.0 (2016-11-03)

- Change method of calculating MLT, see documentation of convert_mlt for details

7.7 1.0.13 (2015-10-30)

- Correctly convert output of subsol() to geodetic coordinates (the error in MLT/mlon conversion was not large, typically two decimal places and below)

7.8 1.0.12 (2015-10-26)

- Return nan in forbidden region instead of throwing exception

7.9 1.0.11 (2015-10-26)

- Fix bug in subsolar/MLT conversion

7.10 1.0.10 (2015-10-08)

- No code changes, debugged automatic build/upload process and needed new version numbers along the way

7.11 1.0.0 (2015-10-07)

- Initial release

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Bibliography

[1] Shepherd, S. G. (2014), Altitude-adjusted corrected geomagnetic coordinates: Definition and functional approximations, J. Geophys. Res. Space Physics, 119, 7501–7521, doi:10.1002/2014JA020264.

[1] Laundal, K. M. and A. D. Richmond (2016), Magnetic Coordinate Systems, Space Sci. Rev., doi:10.1007/s11214-016-0275-y.

Python Module Index

a

Index