sami2py Documentation

Release 0.2.0

Jeff Klenzing

Jul 27, 2020

Contents

CHAPTER 1

Introduction

Sami2py is a python module that runs the SAMI2 model, as well as archives, loads and plots the resulting modeled values. SAMI2 is a model developed by the Naval Research Laboratory to simulate the motions of plasma in a 2D ionospheric environment along a dipole magnetic field [Huba et al, 2000]. SAMI2 solves for the chemical and dynamical evolution of seven ion species in this environment (H^+, He^+, N^+, O^+, $N_2{}^+$, NO^+, and $O_2{}^+$).

The implementation used here includes several added options to the original release of SAMI2. A full list is included in *Modifica*

- The ability to scale the neutral atmosphere in which the ions form through direct modification of the exospheric neutral temperature for extreme solar minimum conditions, as discussed by Emmert et al [2010].
- The ability to switch between HWM93, HWM07, and HWM14 as a user option.

This implementation is based on the MatLab version used in Klenzing et al [2013].

The open-source fortran version of SAMI2 is found at https://www.nrl.navy.mil/ppd/branches/6790/sami2

1.1 How to Cite

When referring to this software package, please cite the original paper by Huba et al [2000] https://doi.org/10.1029/2000JA000035 as well as the package by Klenzing and Smith [2019] https://doi.org/10.5281/zenodo.2875800.

Additionally, please include the following text in the acknowledgements: "This work uses the SAMI2 ionosphere model written and developed by the Naval Research Laboratory."

1.2 References

- Huba, J.D., G. Joyce, and J.A. Fedder, Sami2 is Another Model of the Ionosphere (SAMI2): A new low-latitude ionosphere model, *J. Geophys. Res.*, 105, Pages 23035-23053, https://doi.org/10.1029/2000JA000035, 2000.
- Emmert, J.T., J.L. Lean, and J.M. Picone, Record-low thermospheric density during the 2008 solar minimum, *Geophys. Res. Lett.*, 37, https://doi.org/10.1029/2010GL043671, 2010.

- Klenzing, J., A. G. Burrell, R. A. Heelis, J. D. Huba, R. Pfaff, and F. Simões, Exploring the role of ionospheric drivers during the extreme solar minimum of 2008, *Ann. Geophys.*, 31, 2147-2156, https://doi.org/10.5194/angeo-31-2147-2013, 2013.

CHAPTER 2

Installation

First, checkout the repository:

```
git clone https://github.com/sami2py/sami2py.git
```

Change directories into the repository folder and run the setup.py file. For a local install use the "–user" flag after "install".

```
cd sami2py/
python setup.py install
```

If something has gone wrong, you may be prompted to manually install the fortran executables.

```
make -C sami2py/fortran compile
```

or, on windows,

```
make -C sami2py\fortran compile
```

2.1 Fortran Compilers

By default, sami2py uses gfortran and make to compile the fortran executables. You can get make through pip

```
pip install make
```

If you have a different compiler, you can modify the first line of the fortran Makefile accordingly by using "gf" to point to your compiler of choice. Note that several options are included to ensure the compile is successful.

If you don't have a fortran compiler, gfortran is included as part of the latest gcc package. You can get this from several locations.

For Mac OS X, you can install gcc through package managers such as brew.

For windows, multiple setup options are discussed at https://www.scivision.co/windows-gcc-gfortran-cmake-make-install/

CHAPTER 3

Sample Workflow

In iPython, run:

```
import sami2py
```

If this is your first import of sami2py, it will remind you to set the top level directory that will hold the model output. This should be a string containing the path to the directory you want to store the data in, such as `path='/Users/me/data/sami2py'` or `path='C:\home\data'`. This should be outside the main code directory, so model output files are not confused with model inputs or source code. If you are using Git, it will also ensure that Git does not try to store your local code runs within the repository.

```
sami2py.utils.set_archive_dir(path=path)
```

sami2py will raise an error if this is not done before trying to run the model.

```
sami2py.run_model(tag='run_name', lon=0, year=2012, day=210)
```

Note that the sami2 model runs for 24 hours to clear transients, then begins to output data.

Now load the resultant data:

```
ModelRun = sami2py.Model(tag='run_name', lon=0, year=2012, day=210)
```

The data is stored as *ModelRun.data*, which is an *xarray.Dataset*. Information about the run is stored as 'ModelRun.MetaData', which is a human-readable dictionary of the namelist.

The MetaData can be accessed directly via the dictionary, or through the __repr__. Typing

```
ModelRun
```

yields

```
Model Run Name = test
Day 256, 1999
Longitude = 256 deg
```

(continues on next page)

(continued from previous page)

```
2 time steps from  0.1 to  0.1 UT
Ions Used: H+, O+, NO+, O2+, He+, N2+

Solar Activity
--------------
F10.7: 120.0 sfu
F10.7A: 120.0 sfu
ap: 0

Component Models Used
---------------------
Neutral Atmosphere: NRLMSISe-2000
Winds: HWM-14
Photoproduction: EUVAC
ExB Drifts: Fejer-Scherliess

No modifications to empirical models
```

Full description coming soon

CHAPTER 4

Modifications from SAMI2-1.00

4.1 NRLMSISe-00

This version uses the official release of NRLMSISe-00 (with one modification, discussed below). The unmodified version can be found at https://map.nrl.navy.mil/map/pub/nrl/NRLMSIS/NRLMSISE-00/

The exospheric neutral temperature can be directly scaled by the user for extreme solar minimum conditions, as discussed by Emmert et al [2010]. This is modified by the `Tinf_scale` keyword, and is passed through the namelist as Tinf_scl.

4.2 Photoionization

The photoionization rates can be modified by scaling the resultant EUV spectra. Note that this occurs independently of any modifications to the neutral atmosphere through MSIS. See Klenzing et al [2013] for examples. This is modified by the `euv_scale` keyword, and is passed through the namelist as euv_scl.

4.3 ExB Drifts

Fejer-Scherliess remains the default model for drifts, but the user may now input a series of Fourier coefficients to describe the ExB drifts as a function of local time rather than use the Fejer-Scherliess model. The coefficients are specified by the `ExB_drifts` keyword, which are passed into sami2 through the new exb.inp file. The fourier coefficient routine replaces the sine wave triggered when .fejer. = false.

4.4 Horizontal Wind Model

Sami2py uses the HWM-14 model by default. Users may specify HWM93 or HWM07 for comparison through the `hwm_model` keyword, which is passed through the namelist as hwm_mod.

CHAPTER 5

Contributing

Bug reports, feature suggestions and other contributions are greatly appreciated! Sami2py is a community-driven project and welcomes both feedback and contributions.

5.1 Short version

- Submit bug reports and feature requests through [GitHub](https://github.com/jklenzing/sami2py/issues)
- Make pull requests to the `develop` branch

5.2 Bug reports

When [reporting a bug](https://github.com/jklenzing/sami2py/issues>) 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 *sami2py* for local development:

1. [Fork sami2py on GitHub](https://github.com/jklenzing/sami2py/fork).
2. Clone your fork locally

```
git clone git@github.com:your_name_here/sami2py.git
```

3. Create a branch for local development

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

Now you can make your changes locally. Tests for new instruments are
performed automatically.  Tests for custom functions should be added to the
appropriately named file in ``sami2py/tests``.   If no test file exists,
then you should create one.  This testing uses pytest, which will run tests
on any python file in the test directory that starts with ``test_``.
```

4. When you're done making changes, run all the checks to ensure that nothing is broken on your local system. You may need to install pytest and pytest-flake8 first.

```
pytest -vs --flake8
```

5. Update/add documentation (in `docs`), if relevant
6. 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
```

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

5.5 Pull Request Guidelines

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 an example for use
2. Add a note to `CHANGELOG.md` about the changes
3. Ensure that all checks passed (current checks include Travis-CI and Coveralls)[1]

[1] If you don't have all the necessary Python versions available locally or have trouble building all the testing environments, you can rely on Travis to run the tests for each change you add in the pull request. Because testing here will delay tests by other developers, please ensure that the code passes all tests on your local system first.