Jucker, M 2014 Scientific Visualisation of Atmospheric Data with ParaView. *Journal of Open Research Software,* 2(1): e4, DOI: http://dx.doi.org/10.5334/jors.al

SOFTWARE METAPAPER

Scientific Visualisation of Atmospheric Data with ParaView

Martin Jucker[1]

[1] Program in Atmospheric and Oceanic Sciences, Princeton University, Princeton, USA

The pv_atmos package allows for an automated approach to 4D (3D space + time) visualization of atmospheric data in netCDF format. Using Python scripting, the open source software ParaView is used to load, process, and visualize data. The scripts automatize the loading of data on a longitude-latitude-pressure grid, and re-compute the grid for a log-pressure or spherical representation of all data. Grid lines, planes, and labels can be added if desired. Two example scripts are included in the repository.

Keywords: scientific visualisation; ParaView; Python scripting; geophysical fluid dynamics; netCDF

(1) Overview

Introduction

ParaView [1] is one of the premier open source visualization software packages for 4D (3D space + time) data visualization. It is a very powerful tool built in a very general way to serve visualization needs from many fields. However, with the general-purpose nature of the software, a great deal of data pre-processing and manipulation is often required to create discipline specific visualizations.

This is true in particular for the geophysical fluid dynamics community. ParaView has native support for the network Common Data Form (netCDF), and in particular the widely accepted Climate and Forecast (CF) metadata conventions [2]. However, it provides little built-in functionality, as for instance spherical geometry in pressure coordinates, and the requirement for logarithmic coordinates. In general, the axes functionality is one of the few weak points of ParaView, which has arguably been a hindrance to more widespread usage of the software in the scientific community. The package described in this paper can convert longitude-latitude-pressure gridded data into log-pressure coordinates, and provides the possibility of adding correct axes and notation.

This package does not need a working installation of Python, and no package installation is required. The routines are designed to be run within the Python Shell included in ParaView. As such, it can be used in a supportive way within the GUI, as it builds a fully interactive pipeline in ParaView.

Implementation/architecture

General organization

The package is entirely written in Python, and can be executed within the Python shell of ParaView. Therefore, no installation of any software other than ParaView (not even Python) is needed, as all necessary software is included in any installation.

The package as described here is intended to be a basic starting point for future development. Although the author is at this point developing further additions of geographical data, such as world maps and boundaries, it is the intent of this publication to encourage community based development of pv_atmos.

The package is separated into two scripts: atmos_basic.py and atmos_grids.py. atmos_basic.py contains everything necessary to load a given file, and apply a coordinate transformation in all (3D) directions, including a logarithmic transformation of the z-coordinate (pressure to log-pressure). But the script is not limited to pressure coordinates, as also height (or any other z-coordinate) can be read as well. In addition, functions related to creating a spherical geometry are included.

Even though the netCDF files are assumed to follow the Climate and Forecasting (CF) conventions, the metadata has to follow those conventions only loosely. For instance, the coordinates can have arbitrary names in the netCDF file, as long as the time coordinate has an attribute called "units" containing the word "since".

atmos_grids.py builds on atmos_basic.py, and adds the possibility to add a full 3D grid to the data. In rectangular geometry, one can easily add a box around the data, with labels in all three directions, where the user can choose the positions of the grid lines and labels. In addition, planes can be added to show cross sections along any dimension.

In spherical geometry, the user can add spherical shells. These are cross sections at one given pressure level, and include labelling of the pressure level. In addition, there is the possibility to add any text onto the outermost shell, which can be used as watermark.

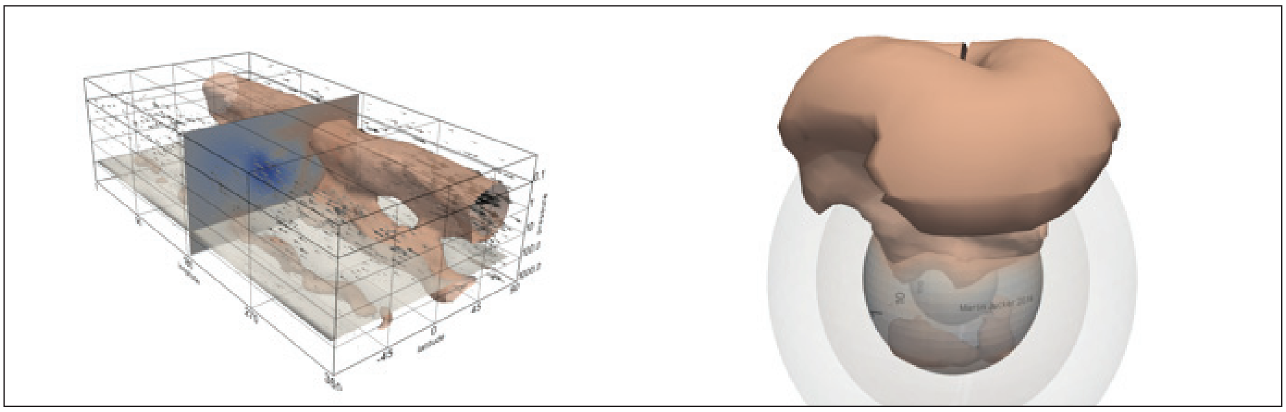

Fig. 1: Result of included scripts example_flat.py (left) and example_sphere.py (right). Shows the zonal wind at a given time step. Included is a customized grid with labels, and cross sections at one given longitude (left) or pressure surface (right).

The example scripts included, example_flat.py and example_sphere.py take data from an example netCDF file, and visualize zonal wind, wind direction and strength, and add grid lines or spherical shells at given pressure levels. Screenshots from these examples are shown in **Figure 1**.

Releases

Releases and versioning is enabled in the GitHub directory, under 'releases'. This folder will contain numerated (versioned) zip files containing the pv_atmos package corresponding to major development steps. It is also planned to make the package available through PyPI.

How to use

All functions contain basic Python docstrings, which can be accessed by typing `help(nameOfFunction)` in the Python Shell.

Both `atmos_basic.py` and `atmos_grids.py` need the math package to be loaded. Thus, before using any of the functions, the user has to invoke `import math`.

Treat data: atmos_basic.py

The most common and basic tasks have been grouped together in the function `loadData`. All input parameters are optional with default values except the input file name. A typical loading command for pressure or height coordinates would be

```
(output_nc, CorrZ,Coor, AspRat) =
loadData(fileName, [coordP,coordY,coordX], 1),
```

or

```
loadData(fileName, [coordZ,coordY,coordX], 0)
```

This will load the file `fileName` (full path including extension) containing the dimensions `[coordP/Z, coordY, coordX]`, and assuming pressure or height coordinates. It will also make sure that pressure assumes positive values, and adjust the aspect ratio to what it is set in the input (default is to keep it as given in the file). After this initial loading, the user can use any of ParaView's GUI or Python interface to further analyse and visualise the data, as the filters `output_nc`, `CorrZ`, `Coor`, and `AspRat` will appear in the pipeline browser as if added from the GUI.

In order to convert to spherical geometry, it is sufficient to invoke the command

```
spherGeom = Cart2Spherical(radius=1, src=AspRat)
```

Note that the camera is not set with those two basic commands, but it can easily be adjusted by clicking the `Reset Camera` button in the ParaView GUI. Again, the user can now revert back to the GUI or python interface to add filters, colors, etc. in the pipeline browser.

Add grid: atmos_grids.py

Again the most common tasks have been grouped together. For rectangular geometries, invoking

```
AddGrid()
```

will add a box around the data, spanning from 0 to 360° in longitude, -90 to +90° in latitude, and 1000 to 0.1hPa in pressure. The default values are changed by adding the appropriate input parameters. Invoking

```
AtmosShells(radius=1, src=CoorZ)
```

will extract planes of constant pressure on the sphere (shells) from the source filter `CoorZ` and show them, labelled, together with the data (see **Figure 1** right).

It is also possible to add slices of data to the visualisation at the appropriate places, and the same for labelling those planes. `AddPresPlane`, `AddLatPlane`, and `AddLonPlane` will add slices at constant pressure (height), latitude (y-coordinate), and longitude (x-coordinate), which are extracted from the data and can be data colored or further used in the visualisation pipeline. In the case of spherical geometry, a subsequent `Cart2Spherical` command will again convert the planes into the spherical geometry.

Note that the `Add???Plane` functions have an input called `data`: if it is set to 1, the cut out plane will be assumed to be within the domain of the data, and a `Slice()` operator will be applied. If `data=0`, the plane will be independent of data, and it is assumed that it will be used as grid lines.

If one likes to add another label to the grid, the functions `AddPresLabel`, `AddLatLabel`, and `AddLonLabel` can be called.

Examples
As an example, the scripts from the pv_atmos package have been used by the author for the winning picture of Princeton University's Art of Science competition in 2013 [3], and one of the finalist movies at the same competition in 2014.
The two examples provided in the repository, `example_flat.py` and `example_sphere.py` load the file `uv_daily.nc` and create a basic visualisation in rectangular or spherical geometry, as shown in **Figure 1**. These examples need ParaView version 4.1 and up to run completely. They also run in earlier versions but don't automatically color the extracted planes and isosurfaces. When running the example scripts, the user first has to set the path to `atmos_basic.py` and `atmos_grids.py`, which is by default ../.

Quality control
The scripts have been successfully tested on all platforms where ParaView is available (Mac, Windows, Linux), with the oldest supported version being 3.14. This version includes Python 2.5.6. The example scripts include a functionality of applying a pre-defined color map through scripting (AssignLookupTable), which is only available for ParaView version 4.1 onwards. Even so, the examples still work on ParaView versions 3.14 and up.

There is also a unit testing routine `unitTest.py` in the folder `testing/`. Note that this file tests the main routines only – these routines call all smaller functions contained in the package. Furthermore, the testing is made using random number generators, so that the pipeline output in ParaView does not necessarily make physical sense.

(2) Availability
Operating system
ParaView 3.14 or up. Refer to www.paraview.org for system requirements. The here discussed scripts have no further requirements.

Programming language
Python 2.5 and up if ParaView module is loaded outside of the ParaView Python shell. Otherwise, Python shipped with ParaView is sufficient, and no programming language has to be installed.

Additional system requirements
Scientific visualization can be demanding on memory and CPU power. However, this depends entirely on the data file. Keeping data files smaller than tens of megabytes should avoid major problems.

Dependencies
Python's math library is needed for the value of pi and for computing logarithms. This is included in ParaView's Python shell. If loaded in independent Python session, paraview.simple needs to be in the Python path.

Code repository
Name
pv_atmos

Persistent identifier
https://github.com/mjucker/pv_atmos.git

License
MIT

Date published
20/02/2014

Language
English

(3) Reuse potential
The pv_atmos package is written in Python and does not need any packages not included with the standard installation of ParaView. It is based on basic ParaView commands, and can be expected to be working without any issues in the future and on any platform. Potential users therefore won't have to worry about the package becoming dysfunctional.

Visualisation of atmospheric data is a growing field, and requires very repetitive treatment of data. In particular, conversion of pressure coordinates into something more easily visualised is a big hindrance to a more wide spread use of visualization software in atmospheric research.
Visualisation of atmospheric data is a growing field, and requires very repetitive treatment of data. In particular, conversion of pressure coordinates into something more easily visualised is a big hindrance to a more wide spread use of visualization software in atmospheric research.

A general hindrance of visualisation software to be useful for scientists is a poor representation of coordinate axes.

The pv_atmos package addresses those two main obstacles for more scientists to use ParaView, while still being very general. In addition, the atmos_grids.py functionality, i.e. the addition of 3D grid lines and labels, can be used even without input data. These functions can therefore be of use to the complete scientific community using ParaView.

As mentioned above, it is the intention of this publication to attract a developer community, such that the package can grow and profit from the needs and skills of a variety of scientists and developers.

References
1. Portal of ParaView. Available at: www.paraview.org
2. Climate and Forecast (FC) conventions. Available at: http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.6/cf-conventions.html
3. Art of Science, Princeton University. Available at: www.princeton.edu/artofscience

How to cite this article: Jucker, M 2014 Scientific Visualisation of Atmospheric Data with ParaView. *Journal of Open Research Software,* 2(1): e4, DOI: http://dx.doi.org/10.5334/jors.al

Published: 3 June 2014