Python tools for EOF analysis and visualization

Liang Feng and Paul Palmer

Uinversity of Edinburgh
June, 2013

I. Introduction

1.1 EOF analysis

Empirical orthogonal function (EOF) analysis is a statistics approach to extract dominate oits simplicity, EOF has been widely used to reduce original data from a many-dimensional vector space to a small set of orthogonal vectors, but without losing much of its variances. EOFs are usually calculated by computing the eigenvalues and eigenvectors of a spatially weighted anomaly covariance matrix of atmospheric variables. Typically a handful of such eigenvectors (i.e, modes) can explain most of the data variability, although whether these orthogonal (independent) modes can be interpreted as atmospheric and oceanographic structures is often matter of controversy.

In this package, we have defined a class *eof_cl (pyeof)* to extract EOF modes from original data by using singular value decomposition (SVD) technique, similar to some existing software packages in several computer languages (see, for examples, *http://www.idlcoyote.com/code_tips/eof_analysis.html*, *http://pmc.ucsc.edu/~dmk/notes/EOFs/EOFs.html*, and *http://ajdawson.github.io/eofs/*)

Members of class *eof_cl* are used to store essential information for decomposition of the original data set. Also it provides member functions for projecting fields to EOF modes, as well as visualizing the EOFs and principal components (PCs) etc.

1.2 Visualization tools

There already exist some powerful python packages for visualizing data sets (for example pylab (http://matplotlib.org), and its toolkit basemap (http://matplotlib.org/basemap/)). Modules in this package are designed to provide easy-use wrappers to these packages to perform some most common visualizations in atmospheric researches, including both the static plots and animations.

In particular, it provides a wxWindow-based (http://www.wxpython.org) GUI module for directly accessing and visualizing BPCH2 outputs from a widely used chemistry transport model GEOS-Chem. Also, the python shell included in this GUI module allows the user to

read in data in other format (such as netCDF) with little efforts, and then visualize them by using functions provided by this GUI (as well as other python packages).

II. Installation

2.1 Download

The package is freely available from website *xweb.geos.ed.ac.uk/~lfeng* as a tar.gz source file.

2.2 Prerequisites

This package is developed as one add-on extension to the PyOSSE toolkit for Observation System Simulation Experiment (OSSE) system developing (see xweb.geos.ed.ac.uk/~lfeng for more details). It is assumed to that PyOSSE package has been installed in the user's system, as it needs some *util* functions (such as *gen_plots.py* for 2D plots*)* included in PyOSSE.

2.3 Unpacking

After downloading the archive file pyeof.tar.gz into your workstation, you can extract the files by using linux command:

tar –zxvf pyeof.tar.gz

As a result, directory called '*py_eof*' will be made. Under it, there are 8 python modules:

- *field_grid.py*
- *images.py*
- *ncfile_open.py*
- *pyeof.py*
- *pygui_view.py*
- *write_3d_bpch.py*
- *wx_map_plot.py*
- *wx_zonalmean_plot.py*

More details on these python modules can be found in Section III.

III. Python modules

3.1 Modules

Table 1 lists main functions of the included python modules. Subsections 3.2 and 3.3 give more information on two top modules *pyeof.py* and *pygui_view.py*, respectively.

Name	Comment	Main Functions/classes
field_grid.py	Read, regrid and process model output data for visualization	1.*get_model_pressure*:calculate pressure at model level from surface pressure 2. *get_model_value*: read values from bpch2 file. 3. *regrid_data*:regrid data 4. *get_model_zonal_mean*: calculate model zonal mean field 5. *get_model_colum*: calculate model column
images.py	Images and icons used by pygui_view.py	1. *getMondrianBitmap* 2. *BitmapFromImage* 3. *getMondrianImage* 4. *getMondrianIcon*
ncfile_open	Simple class for opening and reading NetCDF file.	1. *nc_file_cl* (class)
pygui_view.py	GUI interface for data visualization	See 3.3 for detail
pyeof.py	EOF solver	1.*eof_cl* (class) See 3.2 for details
write_3d_bpch.py	Write 3D-data into a bpch2 binary file	1. *open_3d_bpch2_file*: open one new file to write 2. *write_3d_field*: write 3D data into bpch2 file 3. *close_3d_bpch2_file*: close file
wx_map_plot.py	Frame for animating time series of 2D map data	1.*CanvasFrame* (class)
wx_zonalmean_plot.py	Frame for animating time series of zonal mean data	1.*CanvasFrame* (class)

Table 1. Python modules included in the package.

3.2 pyeof.py

Module *pyeof.py* defines class *eof_cl* as the EOF solver of observed variations. In its initialization function *__init__* , *eof_cl* finds the EOFs for a data set X, defined over a temporal-spatial grid by :

1. Reshaping and filtering bad or missing values in the original data set **X** into one 2D matrix **Z** of shape (*nt, np*), where *nt* is the number of valid data along time direction, and *np* the spatial points.
2. Detrending the columns of the resulting matrix **Z** when required.
3. Using singular value decomposition (svd) to separate **Z** into 3 matrices:

$$\boldsymbol{Z = UWV^t},$$

   where **U** and **V** are two orthogonal matrices, and **W** is diagonal.

4. Calculating EOFs and Principal Components of covariance:

   EOF = **V**, and
   PC=**UW**/(sqrt(*nt*-1)),

   so that the covariance matrix **R**= **PC**^{t} (**PC**)

For convenience, class *eof_cl* also defines other member functions such as:

- *get_pcs*: retrieve principal components
- *get_eofs*: retrieve EOF functions
- *get_eigs*: retrieve eigenvalues for covariance matrix
- *project_field:* project one field to EOFs
- *show_pcs*: plot PCs
- *show_eofs:* plot EOFs
- *show_eof_map*: plot EOF over a map

3.3 pygui_view.py

Module *pygui_view.py* provides a wxWindow-based GUI environment for visualizing climate data. It provides GUIs for directly accessing and visualizing BPCH2 binary data from GEOS-Chem chemistry transport model. Also, its pyshell panel allows the user to read in data in other format, and visualize them using functions integrated *with pygui_view.py* or imported from other python modules.

3.3.1 Graphic User Interfaces

Figure 1 shows the main window used by pygui_view.py.

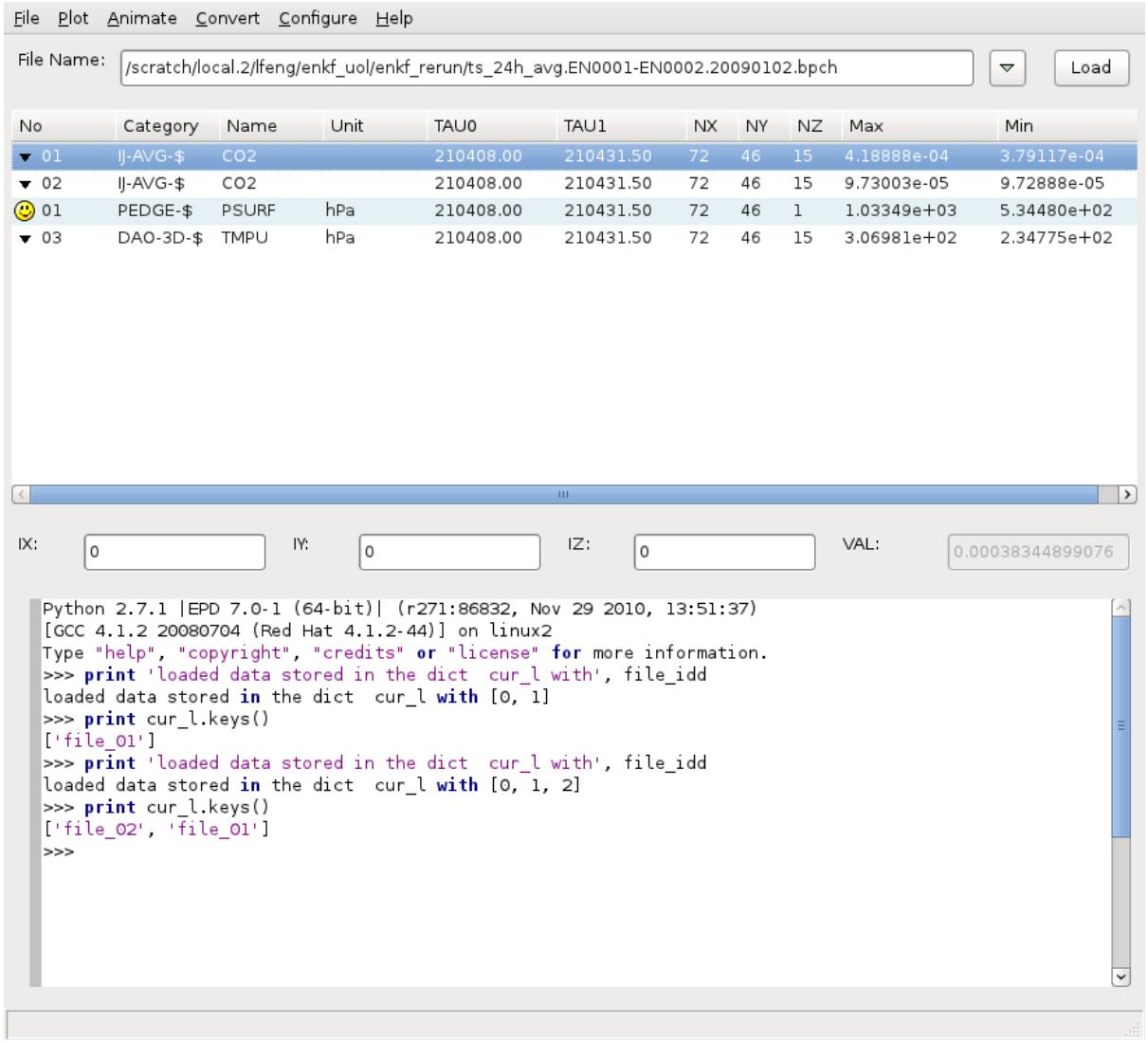

Figure 1. Main window of pygui_view.py

Table 2 lists the objects contained in the main user-interactive window of *pygui_view.py.* The main functions of this application are straightforward and easy to use. More complicated operations can be done through its integrated python shell (see 3.3.2)

Name	Type	Function
File	Menu	menu for operation such as loading file, saving configurations.
Plot	Menu	menu for 2D plots .
--*Total column*	submenu	Make 2D map plot for total columns.
--*Zonal mean*	submenu	Make 2D plot of zonal mean of selected tracer.
--*Map*	submenu	Make 2D map plot at selected model level
--*Option*	submenu	Invoke a dialog window for configuring 2D plots.
Animation	menu	menu for animations
--*Total column*	submenu	Make animation for total columns.
--*Zonal mean*	submenu	Make animation of zonal mean of selected tracer.
--*Map*	submenu	Make animation for selected model level.
--*Option*	submenu	Invoke a dialog window for configuring 2D animation.
Convert	menu	menu for regridding
--*regrid*	submenu	Invoke a dialog for regridding selected field.
Configure	menu	Configurations
File Name	Text box	File name
Browse	Button	Invoke dialog window for file choosing
Load	Button	Load data from BPCH2 file
Fields	list	Information on fields read from BPCH2 file
Python Panel	Panel	Shell for python (see 3.3.2)

Table 2: Objects in main window of *pygui_view.py.*

3.3.2 use of python shell

The python panel integrated with *pygui_view.py* is a standard IDLE python shell, but with many pre-imported modules. So it can be used to execute any python commands. Also, it can be used to read in data in other format for a quick visualization. For example, only 4 sentences are needed to read in latitude (lats), longitude (lons) and surface pressure (sp) from a netCDF file 'sp.nc', and show it in a 2D map plot:

1. sp_f=n4fc.ncfile_cl() # which invokes a dialog to pick netcdf file
2. sp_f.listvar() # list variables
3. lons, lats, sp=sp_f.getvar(['lons', 'lats', 'sp'])
4. gpl.plot_map(sp, lons, lats)

From this panel, the user can also invoke *wx_map_plot.py* and *wx_zonalmean_plot.py* to animate time series of model outputs.

References:

Thompson, D. W. J., and J. M. Wallace, 1998: The Arctic Oscillation signature in wintertime signature in wintertime geopotential height and temperature fields. Geophys. Res. Lett, 25, 1297-1300.