A Project Report

On

ADVANCEMENT OF EOF-BASED GUI

BY

ATMESH MAHAPATRA

2019B4A30560P

Under the supervision of

DR. SUMANTA PASARI

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF

MATH F376: DESIGN PROJECT

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)

PILANI CAMPUS

(DEC 2022)

CONTENTS

Introduction

EOF (Empirical Orthogonal Function) Analysis, is a commonly used technique in statistics since its' invention in the late 1940s.

The method can compress the main dataset into several orthogonal basis vectors. It is sufficient to characterize the intrinsic characteristics of the data. Hence, it is a widely used technique that serves various purposes like compressing the data, detection study, and linear estimation study. EOF is a widely used technique in various disciplines like climate science, oceanography, and meteorology, as it can serve various purposes.

The GUI's (Graphical User Interface) primary aim is to make the implementation of the EOF Analysis easier for anyone, as they won't have to code out the algorithm themselves, and they can simply upload the dataset onto a GUI, and select from the various options available, of the type of EOF, method of computation of EOF, or to obtain data related to the EOF of the dataset, and get the results presented to them.

In addition to the effort it saves, it also saves a lot of time, as the users won't have to bother with going out of their way to understand the derivation and workings of the algorithm, and then spend time implementing it. This is a simple-to-use application.

Having an EOF-based GUI has various other advantages as well. (First, it helps us in seeing the various representations). But more importantly, it far simplifies and speeds up the task of calculating the EOFs. While regular EOF is most commonly used and is the easiest to implement, it often underperforms compared to the other EOFs in certain situations.

In the last project, the basics of GUI had been built, however, to enhance its' utilities, some more features were required. This project aims at covering that gap, so that it can be published in the academic community as an essential tool to help researchers analyze data.

Literature Survey

EOF can be done in multiple ways, but they all have similar underlying principles.
We shall first look at the computation of EOF using the data covariance matrix, as it is the most fundamental way of calculating EOF, and helps us understand the "why" of EOF.

Since EOF is a technique that has been around for various decades, and with ever-evolving need to study various kinds of data, its usage has only increased over the years. Presently, EOF has evolved to have various methods of computing as well as various types that work well in different situations. EOF can be calculated by computing the data covariance matrix, or by Singular Value Decomposition, or by other decomposition methods. The most widely used decomposition techniques are Eigendecomposition and LU Decomposition.

Within EOF, there are multiple techniques of EOF, and multiple types of EOF, each of which usually has a different result than the others. EOF can be calculated using SVD, calculated using Covariance, and calculated using Correlation. The commonly used EOF types are, Regular EOF, Rotated EOF, Complex EOF, and Extended EOF. There are other types as well, like Cyclostationary EOF and Periodic Extended EOF, but they are beyond the scope of the GUI.

The different types of EOF work best in different kinds of scenarios. For instance, regular EOF is most frequently used in general Spatiotemporal datasets, but it has its limitations, its interpretation is often challenging owing to its orthogonality constraint, and most physical systems are not necessarily orthogonal.
To counter such limitations, Rotated EOFs (REOF) are used. As the name suggests, we just "Rotate" the EOFs. The most common rotation technique, and the rotation technique used in the EOF, is the Varimax rotation. It solves the issue in a way since it is more interpretable than regular EOF, but the making the bigger loadings bigger, and the smaller loadings smaller. In REOFs, another approach is oblique rotation, but they preserve neither the orthogonal components nor the PCA, hence, oblique rotation isn't used.
Extended EOFs (EEOFs) are often used as it accounts for temporal correlations in addition to the spatial correlations (which are found in Regular EOFs). This nature of EEOF makes it a useful tool in the study of climatic data, or other types of datasets where there is some kind of seasonality or cyclicity.
Complex EOFs are better than the above EOFs when it comes to identifying moving patterns.

“eofs” is a python library, with ready made EOF functions that makes it easier to compute the EOF for a dataset using various techniques. The GUI contains these functions, and the user can load csv files onto the GUI, and freely apply the functions, instead of writing multiple programs for different kinds of methods of EOF calculation.

This is why it is important to build a GUI that not only makes it easier to perform EOF Analysis on a dataset, but also gives the user more options in implementation.
To achieve this, I went through various papers on professional GUI used in research, and found various ways of improving the “look and feel” of the GUI to suit it for professional use.

Implementation and Methodology

For computing EOF by SVD, the steps are significantly less, if M is our dataset matrix, we take the SVD of MM^T ,

$$MM^T = U\Sigma V^T$$

Here, U contains the eigenvectors, and Σ contains the corresponding eigenvalues **.** When we project U onto M, we get the EOFs of M. The eofs library primarily uses this technique to compute the EOF of a dataset.

In addition to regular EOF, the GUI enables the users to implement Rotated, Complex and Extended EOFs under the "Types of EOF" section, which is not a part of the EOFs library. This makes the implementation of EOF much easier for the user, as they can look at the performance of different types of EOF, and choose the one which accounts for the most variation.

In Rotated EOF, there are 2 types of rotation, Orthogonal Rotation and Oblique Rotation, in Orthogonal Rotation we perform varimax rotation on EOFs to make the larger loading larger, and smaller loadings smaller, thereby, minimizing the mode complexity. We rarely perform Oblique Rotation, as it preserves neither the orthogonality of the modes, nor of the Principal Components. However, Oblique Rotation is carried out by performing Quartimax or Quartimin rotation.

For Complex EOF, we convert the dataset to a complex variable dataset. We do this by taking the Hilbert Transform of the original dataset, which becomes the imaginary part of the complex variable, with the original part being the real part of the variable. Now that we have the complex variable dataset, we find the eigenvectors and eigenvalues of the covariance matrix of the complex variable. This can be done using SVD or the usual method of finding eigenvectors.

In extended EOF, we take a time lag, and try to maximize the variation in a $n_x \times n_y \times n_l$ window (where x denotes rows, y denotes columns, and lags is denoted by l). Let $\overline{x_i}$ be data for station i , then,

$X' = [\{\overline{x_1} .. \overline{x_T}\} ... \{\overline{x_{T-L}} .. \overline{x_T}\}] = [\overline{x'_1} .. \overline{x'_T}]$ where $\overline{x'_i} = \{\overline{x_i} \dots \overline{x_{i+L}}\}$

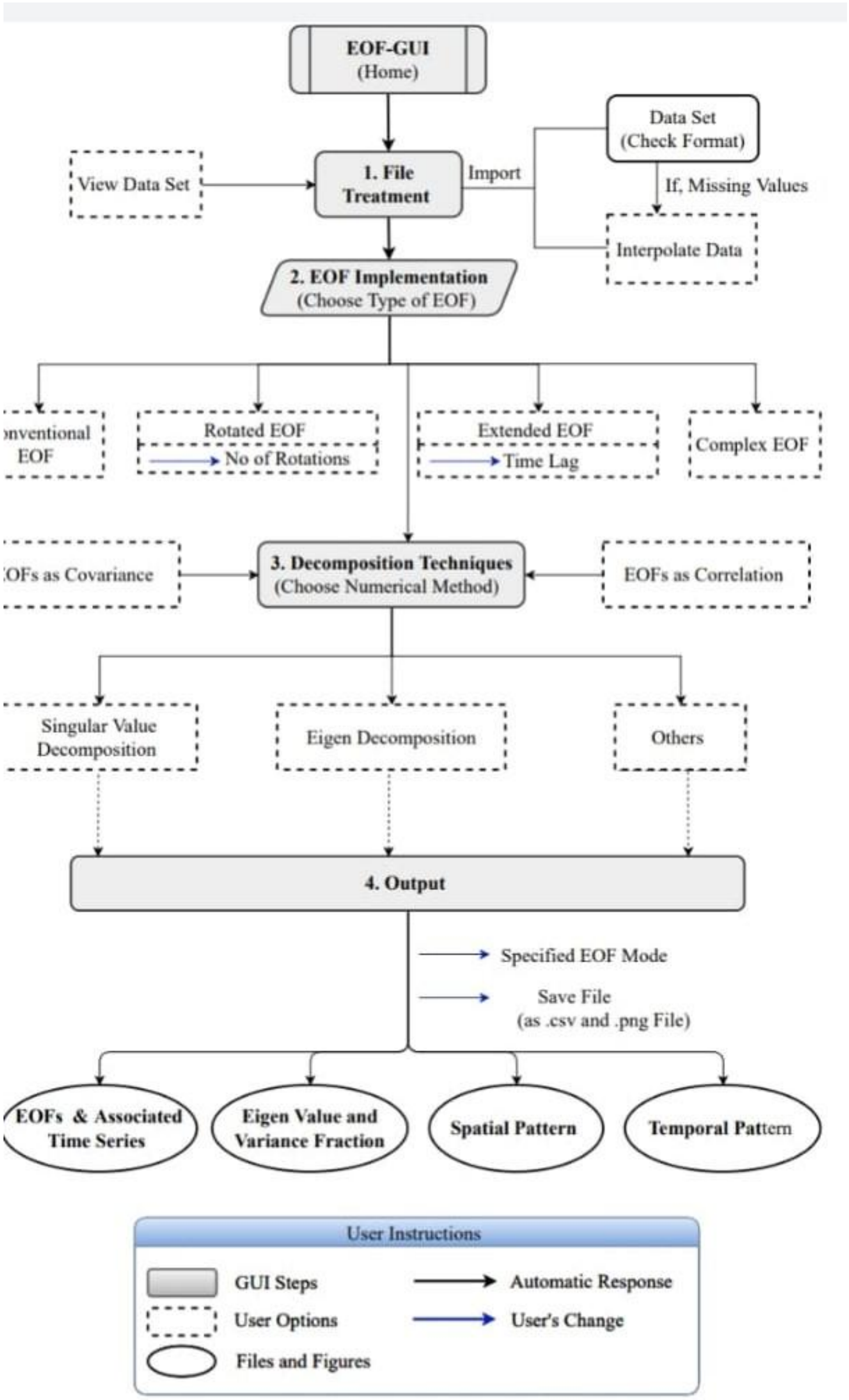
And then, we proceed as we do in regular EOF.

Other than the type of EOF, the user is given a choice of using different decomposition techniques to calculate the EOF, the user can use Eigendecomposition, LU Decomposition or the usual SVD. We perform the decomposition on MM^T and always take the left matrix that is obtained as it contains the eigenvector, and then, we take the projection of the left lector on the dataset.

The GUI is built using Tkinter, which is a Python library used for making GUI. The library lets us build a GUI for any operating system, and provides the programmer with the ability to add standard widgets to their GUI. In our case, the usage of common labels, buttons and textboxes, enables us to complete our GUI.

For the new features in plotting, we have used the cartopy and matplotlib libraries, and the demonstration of the GUI has been done using Jupyter notebook.

The GUI has been built in accordance to the Flowchart in the following page.

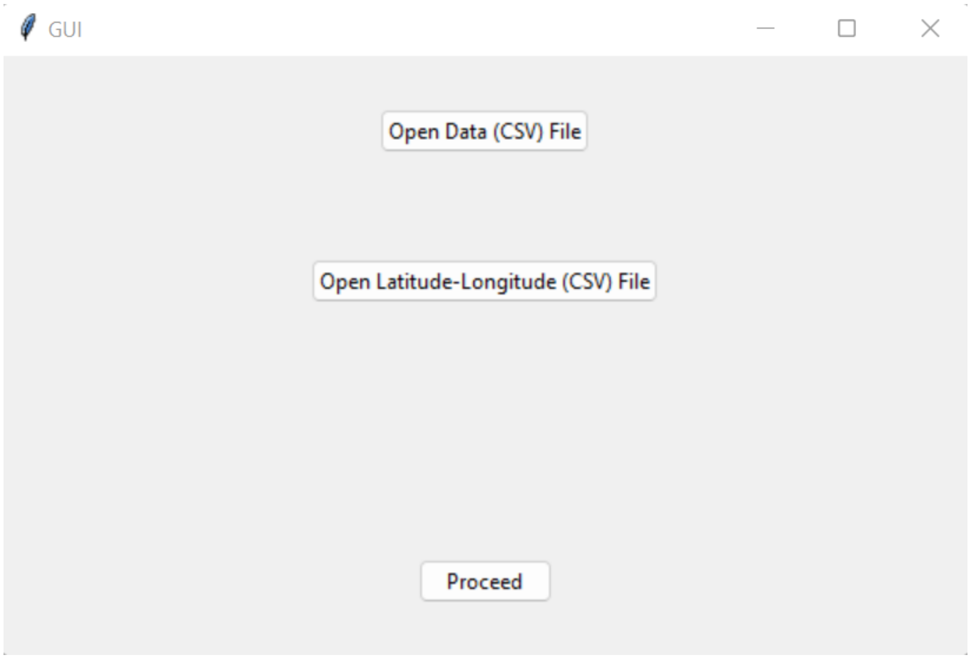

EOF-GUI
(Home)
View Data Set
1. File Treatment
Import
Data Set
(Check Format)
If, Missing Values
Interpolate Data
2. EOF Implementation
(Choose Type of EOF)
onventional EOF
Rotated EOF
No of Rotations
Extended EOF
Time Lag
Complex EOF
OFs as Covariance
3. Decomposition Techniques
(Choose Numerical Method)
EOFs as Correlation
Singular Value Decomposition
Eigen Decomposition
Others
4. Output
Specified EOF Mode
Save File
(as .csv and .png File)
EOFs & Associated Time Series
Eigen Value and Variance Fraction
Spatial Pattern
Temporal Pattern
User Instructions
GUI Steps
User Options
Files and Figures
Automatic Response
User's Change

The Final GUI Manual

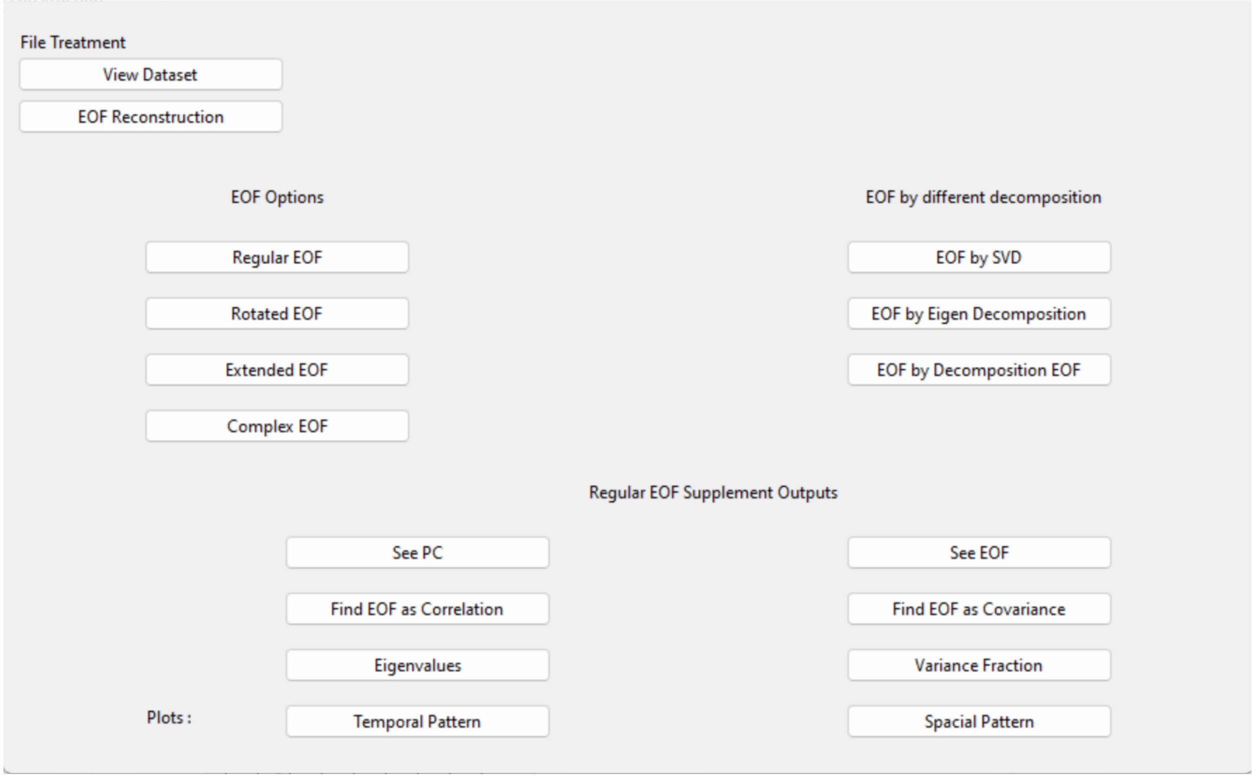
First, the user must download the python file, and copy the contents of the code onto Jupyter notebook. The user must install the following required libraries:-

1. EOFs library
2. Cartopy
3. SciPy
4. Matplotlib
5. Other inbuilt libraries (if the user does not have one): Numpy, Pandas, Tkinter.

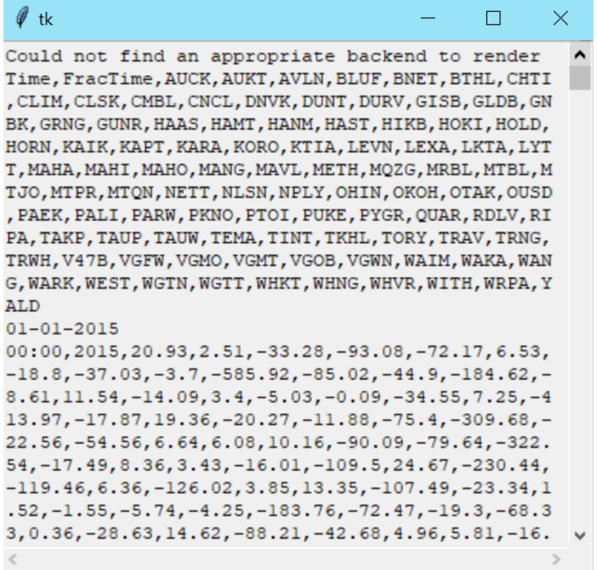
The first window that appears is:



As written in the button, one can select the csv file that contains the users' dataset. On clicking on "Proceed" the user is taken to:



View Dataset:
The user can use “View Dataset” to check the dataset they have uploaded.
Then, the user can choose whether they want to implement the Regular EOF, or the different types of EOF, or the different decomposition techniques.



```
Could not find an appropriate backend to render
Time,FracTime,AUCK,AUKT,AVLN,BLUF,BNET,BTHL,CHTI
,CLIM,CLSK,CMBL,CNCL,DNVK,DUNT,DURV,GISB,GLDB,GN
BK,GRNG,GUNR,HAAS,HAMT,HANM,HAST,HIKB,HOKI,HOLD,
HORN,KAIK,KAPT,KARA,KORO,KTIA,LEVN,LEXA,LKTA,LYT
T,MAHA,MAHI,MAHO,MANG,MAVL,METH,MQZG,MRBL,MTBL,M
TJO,MTPR,MTQN,NETT,NLSN,NPLY,OHIN,OKOH,OTAK,OUSD
,PAEK,PALI,PARW,PKNO,PTOI,PUKE,PYGR,QUAR,RDLV,RI
PA,TAKP,TAUP,TAUW,TEMA,TINT,TKHL,TORY,TRAV,TRNG,
TRWH,V47B,VGFW,VGMO,VGMT,VGOB,VGWN,WAIM,WAKA,WAN
G,WARK,WEST,WGTN,WGTT,WHKT,WHNG,WHVR,WITH,WRPA,Y
ALD
01-01-2015
00:00,2015,20.93,2.51,-33.28,-93.08,-72.17,6.53,
-18.8,-37.03,-3.7,-585.92,-85.02,-44.9,-184.62,-
8.61,11.54,-14.09,3.4,-5.03,-0.09,-34.55,7.25,-4
13.97,-17.87,19.36,-20.27,-11.88,-75.4,-309.68,-
22.56,-54.56,6.64,6.08,10.16,-90.09,-79.64,-322.
54,-17.49,8.36,3.43,-16.01,-109.5,24.67,-230.44,
-119.46,6.36,-126.02,3.85,13.35,-107.49,-23.34,1
.52,-1.55,-5.74,-4.25,-183.76,-72.47,-19.3,-68.3
3,0.36,-28.63,14.62,-88.21,-42.68,4.96,5.81,-16.
```

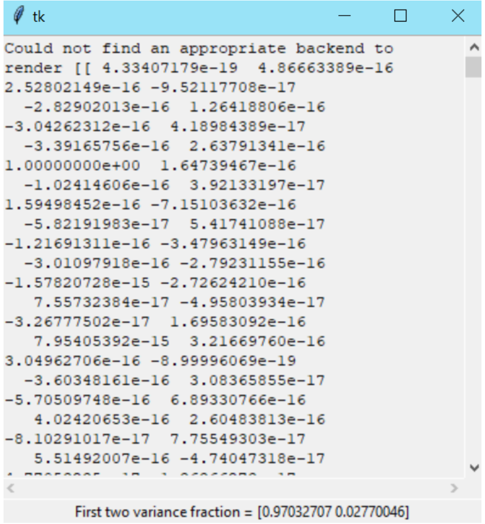
Above is the GPS dataset that was uploaded onto the GUI.

PC shows the Principal Component of the dataset, and EOF shows the EOF. One might wonder that PC and EOF are the same, but that is the case if we use the data covariance method. In SVD, we get a left matrix and a right matrix. The projection of the left matrix onto the original dataset gives us the EOF, and the projection of the right matrix gives us the PC. All of these options in this window is a direct implementation of methods in eofs library, which describes its' functions as:

pcs - The (optionally scaled) principal component time series (PCs).
eofs - The (optionally scaled) empirical orthogonal functions (EOFs).
eofsAsCorrelation - The EOFs expressed as the correlation between each PC and the input data set at each grid point.
eofsAsCovariance - The EOFs expressed as the covariance between each PC and the input data set at each grid point.
eigenvalues - The eigenvalues (decreasing variances) associated with each EOF mode.
varianceFraction - The fraction of the total variance explained by each EOF mode.

Types of EOF option:

Here, the user can implement Regular, Rotated, Extended, or Complex EOF, and directly obtain the results, and hence, choose which EOF to use for their analysis.
As an example:



```
Could not find an appropriate backend to
render [[ 4.33407179e-19  4.86663389e-16
2.52802149e-16 -9.52117708e-17
  -2.82902013e-16  1.26418806e-16
-3.04262312e-16  4.18984389e-17
  -3.39165756e-16  2.63791341e-16
1.00000000e+00  1.64739467e-16
  -1.02414606e-16  3.92133197e-17
1.59498452e-16 -7.15103632e-16
  -5.82191983e-17  5.41741088e-17
-1.21691311e-16 -3.47963149e-16
  -3.01097918e-16 -2.79231155e-16
-1.57820728e-15 -2.72624210e-16
   7.55732384e-17 -4.95803934e-17
-3.26777502e-17  1.69583092e-16
   7.95405392e-15  3.21669760e-16
3.04962706e-16 -8.99996069e-19
  -3.60348161e-16  3.08365855e-17
-5.70509748e-16  6.89330766e-16
   4.02420653e-16  2.60483813e-16
-8.10291017e-17  7.75549303e-17
   5.51492007e-16 -4.74047318e-17
```

First two variance fraction = [0.97032707 0.02770046]

The above is the result obtained by clicking on "Rotated EOF". It shows the final rotated EOF, and the associated variance fraction with the first two modes.

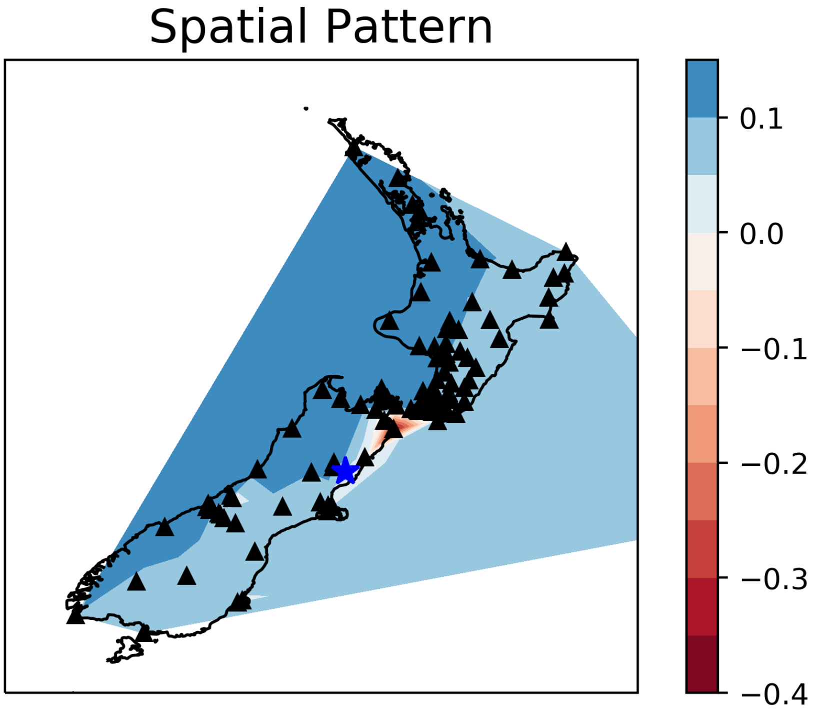
EOF by different decomposition techniques:

One can choose the decomposition technique to use - SVD, Eigen Decomposition, LU Decomposition.

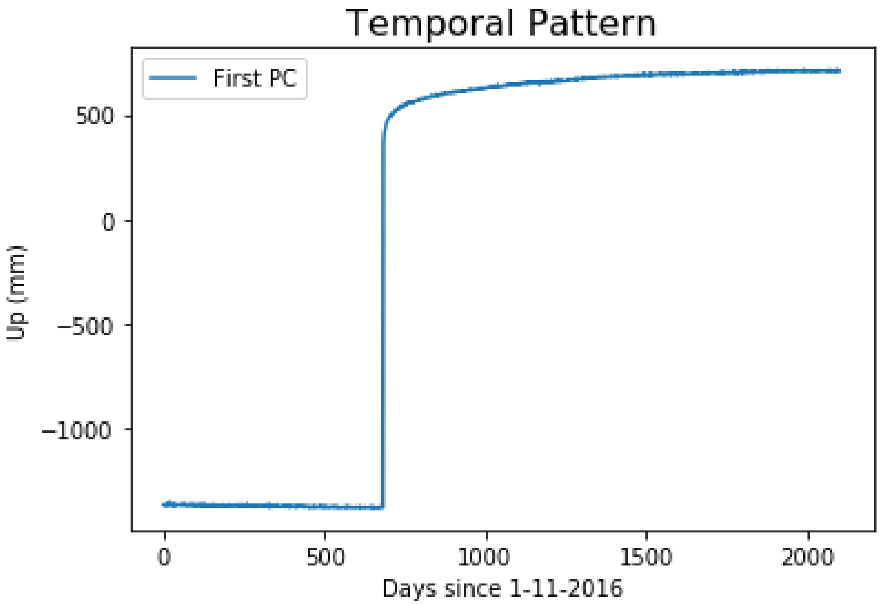
The project is primarily built for Spatio-temporal data (As is often encountered for studying crustal deformation), like GPS data, on which the GUI has been tested. However, in most cases, it should also work for other types of 2-dimensional data.

Plots:
One can also obtain the spatial and temporal plots as shown in the figures below:

The first dataset used is the GPS Dataset for New Zealand to Analyze Crustal Deformation.
Spatial Pattern (First mode of EOF):

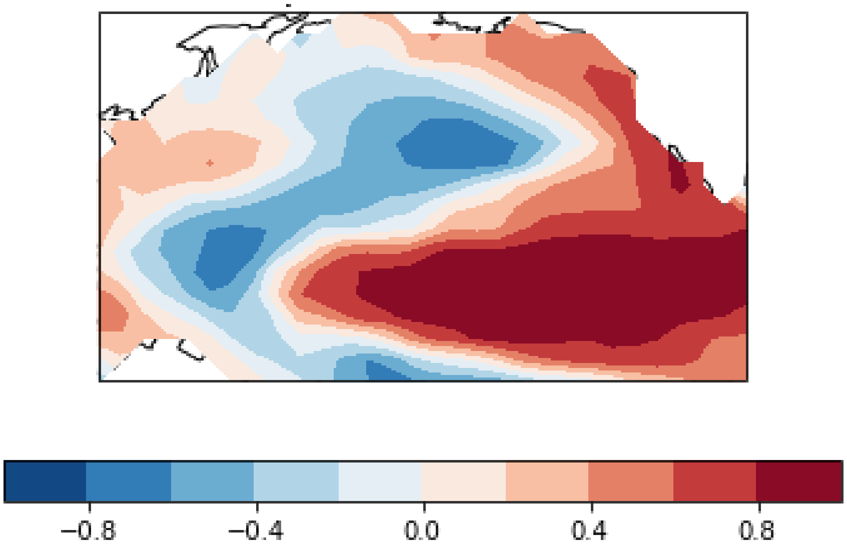


Temporal Pattern (1st principal component):

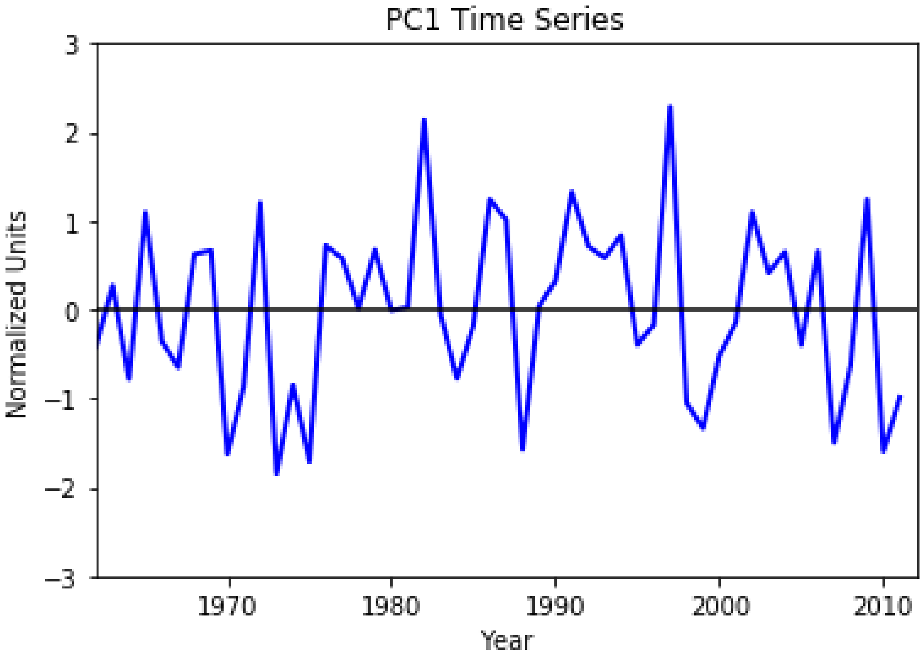


Second Dataset, Atmospheric Pressure Dataset - Study of El Nino:-

Spatial Pattern (EOF):-

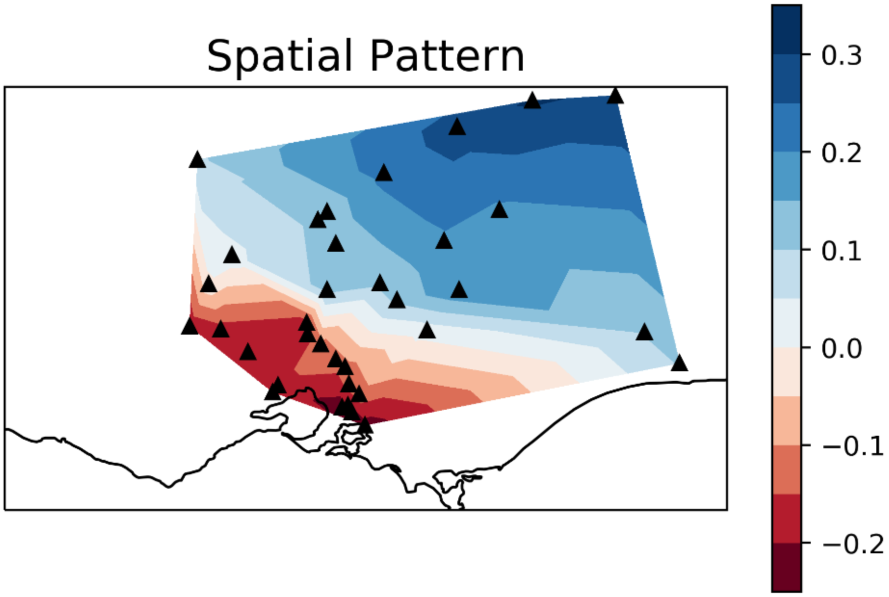


Temporal Pattern (1st PC):

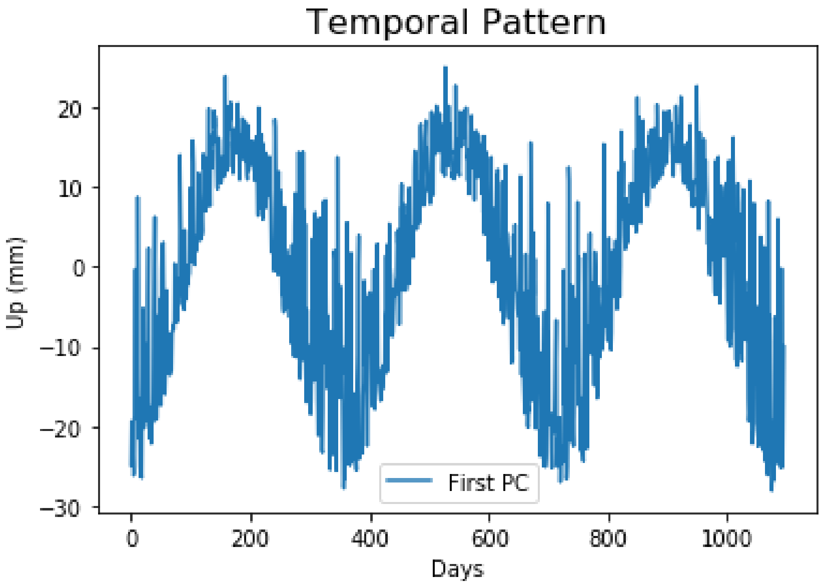


Third Dataset, GHI Dataset - South Australia:-

Spatial Pattern (EOF):-



Temporal Pattern (1st PC):-



Conclusion and Future Work

The GUI allows the users to compute EOF, and gives a lot of freedom to the user about the techniques they wish to implement.

However, there are a few issues with the GUI. It looks very basic, and there is scope of making it more usable, for instance, there can be a functionality of exporting the EOF to a csv file, or just to make its functioning seamless.

For this purpose, the application and code is publicly available for anyone to edit according to their preferences.

Open Source Repository of the GUI: https://github.com/atmeshvm/EOF-based-GUI

References

1. S.A. Bjornsson and S.A Venegas (1997): A Manual of EOF and SVD analysis of Climatic Data
2. Emmy T.Y. Chang and Benjamin F. Chao (2014): Analysis of coseismic deformation using EOF method on dense, continuous GPS data in Taiwan
3. Kwang-Y. Kim AND Qigang Wu (1997): A Comparison Study of EOF Techniques: Analysis of Nonstationary Data with Periodic Statistics
4. Peng Chen, Lixia Liu, Yibin Yao (2020): A Global EOF model of Plasmaspheric Electron Content
5. A. Hippert-Ferrer, Y. Yan, P. Bolon (2019): Gap-filling based on iterative EOF analysis of temporal covariance
6. Danang Eko Nuryanto (2016): A complex empirical orthogonal function for combining two different variables over the Indonesian maritime continent
7. J.D. Horel (1984): Complex Principle Component Analysis: Theory and Examples.
8. Florian Gerber, Rogier de Jong, Michael E. Schaepman (2018): Predicting Missing Values in Spatio-Temporal Remote Sensing Data.
9. Alexander Barth, Robert Weisberg (2007): Correction to "Multivariate reconstruction of missing data in sea surface temperature, chlorophyll, and wind satellite fields"
10. Dawson, A. (2016): eofs: A Library for EOF Analysis of Meteorological, Oceanographic, and Climate Data. *Journal of Open Research Software*, 4(1), p.e14. DOI: http://doi.org/10.5334/jors.122
11. A. Hannachi,* I. T. Jolliffe and D. B. Stephenson (2016): Review Empirical orthogonal functions and related techniques in atmospheric science: A review