NEROGRAV

New Refined Observations of Climate Change from Spaceborne Gravity Missions

International Spring School

10.-14. March 2025

Anne Springer, Mail: springer@geod.uni-bonn.de Christian Mielke, Mail: mielke@geod.uni-bonn.de

Practical 3: Global analysis of gridded total water storage data

Purpose of the practical: In this practical the GRACE time series of global total water storage anomalies will be analyzed for different (geophysical) signals. First, trends and amplitudes of global GRACE grids will be investigated. Then, the method of "Principal Component Analysis" (PCA) will be applied to time series of GRACE data and to hydrological model output provided by the GLDAS-NOAH model. PCA is used to extract individual dominant *modes* of the data variability, while simultaneously suppressing those modes connected with low variability. The given time-space data field (e.g. monthly data given on a geographical grid) is separated into spatial structures called *empirical orthogonal functions* (EOF) and their amplitudes in time, called *principle components* (PCs). PCA reduces the dimension of data efficiently.

Additional material: Lecture Notes "Analysis Tools for GRACE- and Related Data Sets" by Jürgen Kusche et al., "Summerschool Global Hydrological Cycle", Mayschoss, Sept. 12-16, 2011.

Directories: The data sets needed for this practical can be found in the directory *data_Lab34*. The provided matlab functions can be found in the directory *functions_Lab34*. The provided python functions can be imported from the file `SpringSchoolLib34.py`.

Exercise 1: Investigating trends and periodic signals

1. Load the file `grace_tws.mat` (Matlab: `load()`, Python: `from scipy.io import loadmat`) to obtain a time series of gridded total water storage anomalies (TWSA) expressed in equivalent water heights (ewh) based on GRACE observations. The file contains longitude (lon, $n \times 1$), latitude (lat, $m \times 1$), time ($time\_ewh\_grace$, $t \times 1$) and GRACE derived ewh (ewh_grace, $n \times m \times t$, [mm]) filtered with the DDK3 filter and evaluated on a 1° grid.

2. Apply the function `time2decimalYears` to convert the temporal information of the GRACE data to decimal years.

3. Compute trend, annual- and semiannual signal using the function `fitTrend2D` and visualize the results using the function `showData`. The amplitude of the annual (semiannual) signal can be computed from corresponding sine and cosine parameters according to $A = \sqrt{p_{sin}^2 + p_{cos}^2}$.

- Which is the unit of the computed trend?
- How does the trend change when selecting different time spans from the GRACE data?
- Which geophysical signals explain the computed trends and amplitudes?
- optional: How does the annual amplitude change when applying different filters to the GRACE data? (please use your functions from practical 2 to generate differently filtered time series.)

Exercise 2 (optional): Regression against climate indices (Niño 3.4)

1. Load the file `index.mat`. The first column contains years, the second column months, and the third column the Niño 3.4 indicator of central tropical Pacific sea surface temperature anomalies.
2. Apply the function `time2decimalYears` to convert the temporal information of the index data to decimal years.
3. Apply the function `fitIndex` to perform a multilinear regression of GRACE data against the Niño 3.4 index. Make sure that you select a consistent time span.
4. Visualize the gridded amplitudes of the index (`showData`) to provide an estimate of the magnitude of the total influence of the El Niño phenomen on terrestrial water storage.

Exercise 3a: Calculation and visualization of EOFs and PCs

1. Load the file `noah_tws.mat`. The file contains a monthly time series of filtered (DDK3) total water storage anomalies from the GLDAS-NOAH model (*ewh_noah* [mm]) on a $1^\circ \times 1^\circ$ geographical grid and corresponding temporal information (*time_ewh_noah*). The model output is provided on the same geographical grid like the GRACE data used in Exercise 1.
2. Select the time span 2003 to 2016 from GRACE and GLDAS-NOAH.
3. Interpolate missing GRACE data using
   - Matlab: the Matlab-intern function `interp1` (Hint: this is possible without any for-loop).
   - Python: the SciPy function `interp1d` (`from scipy.interpolate import interp1d`).
4. Prepare the data for principal component analysis (PCA) by sorting the gridded values of each month into one column of the matrix $\mathbf{Y}$ (longitude-wise). Thus, the data matrix $\mathbf{Y}$ has the dimension $p \times t$ with p = number of grid points and t = number of points in time.
5. Calculate the spatial patterns (EOFs) from the GLDAS-NOAH data set using the function `calculateEOF`. These patterns serve as basis functions for further calculations.

6. Visualize the spatial patterns for the first four modes using the function `showEOF`.

7. Calculate the temporal evolution (PCs) of the above calculated GLDAS-NOAH basis functions both from the GRACE and the GLDAS-NOAH data using the function `calculatePC`.

8. Visualize the temporal evolution for the first four modes, both for GRACE and GLDAS-NOAH within one figure using the function `showPCs`.

9. Compare and interpret the results obtained from GRACE and GLDAS-NOAH.

Exercise 3b: Understanding compression properties of PCA

1. Visualize the eigenvalues calculated in Exercise 3a. You can use the function `showEigenvalues` for an easy visualization.

2. How many modes are needed to reconstruct 80% and 95% of the GLDAS-NOAH variability? Calculate the fraction of the signal variability reconstructed by $\bar{p}$ modes according to

$$var_{\bar{p}} = \frac{\sum_{j=1}^{\bar{p}} \lambda_j}{\Delta^2} \qquad \text{with the total variance} \qquad \Delta^2 = \sum_{j=1}^{p} \lambda_j. \tag{1}$$

3. From a given matrix $\mathbf{E}$ containing the EOFs in its columns and a matrix $\mathbf{D}$ containing the PCs in its rows, the signal matrix $\mathbf{Y}$ can be reconstructed by

$$\mathbf{Y} = \mathbf{ED}. \tag{2}$$

   Use the calculated PCA from the globalGLDAS-NOAH time series to reconstruct 80% and 95% of the variability.

4. Plot the reconstructed signal (80%, 95%) and the original signal and the difference between both using the function `showData` for one arbitrary month (e.g. 2005-06).

Exercise 3c (optional): Understanding domain dependence of PCA

1. Use the global GLDAS-NOAH time series.

2. Cut out the data in the Amazon region and in the Orinoco region using the function `selectPolygon`. The boundary polygons for the two regions are provided by the files `amazon.mat` and `orinoco.mat`. Each file contains a matrix with two columns, the first column consisting of the longitude values of the polygon points, the second column containing the latitude values.

3. Compute the EOFs and PCs for each region separately using the functions `calculateEOF` and `calculatePC`.

4. Visualize only the first EOF and PC for each region. Here you can use the function `showEOFlocal`.

5. How many modes would be necessary to reconstruct 95% of the signal?

6. Compare the regional results and the global results (that you obtained from Exercise 3a) and discuss.

Functions: Variables marked in italic are optional.
Functions for visualization:

function showData (data,longitude,latitude,titleString,*label,clim,cmap,reverse*)	
Visualization of globally gridded data.	
Input	• `data`: $n \times m$ vector containing the data values. • `longitude`: $n \times 1$ vector containing the longitude values [degree]. • `latitude`: $m \times 1$ vector containing the latitude values [degree]. • `titleString`: string containing the title of the Figure. • `label`: string containing the unit of the data. • `clim`: 2×1 vector containing the lower and upper limit of the colorbar, e.g. [-100 100] • `cmap`: choice of colormap: e.g. jet, parula • `reverse`: boolean: if reverse==1 invert colormap
Output	• 2D-plot of global gridded values

function showEigenvalues (Eigenvalues, titleString)	
Visualization of the evolution of the eigenvalues.	
Input	• `Eigenvalues`: $p \times 1$ vector of eigenvalues. • `titleString`: a string describing the data.
Output	• Plot of the eigenvalues.

function showEOF (EOF, longitude, latitude, i, titleString)	
Visualization of the spatial pattern of one specific EOF with the index i.	
Input	• `EOF`: $n * m \times 1$ vector containing the gridded values of the one EOF to be plotted, e.g. one column of the matrix $\mathbf{E}$. • `longitude`: $n \times 1$ vector containing the longitude values of the grid points. • `latitude`: $m \times 1$ vector containing the latitude values of the grid points. • `i`: index of the EOF to be visualized (EOFs are sorted according to size of eigenvalue), needed for title of the plot. • `titleString`: a string describing the data source (e.g. "GLDAS-NOAH"), used for the title of the plot.
Output	• Plot of the spatial pattern.

function showEOFlocal (EOF, longitude, latitude, border, i, titleString)	
Visualization of the spatial pattern of one specific EOF with the index i on a spatial domain limited by the polygon `border`. The dimension n refers to the number of grid points within the polygon.	
Input	• `EOF`: $n \times 1$ vector containing the gridded values of the one EOF to be plotted. E.g. one column of the matrix $\mathbf{E}$. • `longitude`: $n \times 1$ vector containing the longitude values of the grid points. • `latitude`: $n \times 1$ vector containing the latitude values of the grid points. • `border`: $b \times 2$ matrix which contains the polygon of a region (e.g. amazon). The first column consists of the longitude values of the polygon points, the second column contains the latitude values. • `i`: number of the EOF to be visualized (EOFs are sorted according to size of eigenvalue), needed for title of the plot • `titleString`: A string describing the data source (e.g. "GLDAS-NOAH"), used for the title of the plot.
Output	• Plot of the spatial pattern for a regional area.

function showPCs (PCs, time, i, legendStrings)	
Visualization of the principle component with the index i.	
Input	• `PCs`: $p \times u$ vector containing the principle components which shall be visualized, e.g. the i-th row of the matrix $\mathbf{D}_{grace}$ and of matrix $\mathbf{D}_{noah}$ ($[PCs = [PCi_{grace} PCi_{noah}]$). u is the number of components displayed, so u = 2 if the first PC is displayed for GRACE and NOAH. • `time`: $p \times 2$ vector containing time in the format [years months] • `i`: number of the PC to be visualized (sorted according to the size of the eigenvalues), needed for title of the plot. • `legendStrings`: legend entries provided as a struct of strings, e.g. {'GRACE', 'NOAH'}
Output	• Plot of the principle component.

Functions for computation:

function [time] = time2decimalYears (years, months)	
Convert time provided as vectors of years and months to decimal years	
Input	• `years`: $p \times 1$ vector of years • `months`: $p \times 1$ vector of months
Output	• `time`: $p \times 1$ vector of time in decimal years

function [IN, lon, lat] = selectPolygon (longitude, latitude, polygon)
Select longitudes and latitudes located within a polygon

Input	• `longitude`: $n \times 1$ vector containing the longitude values of the grid points. • `latitude`: $m \times 1$ vector containing the latitude values of the grid points. • `border`: $b \times 2$ matrix which contains the polygon of a region (e.g. amazon). The first column consists of the longitude values of the polygon points, the second column contains the latitude values.
Output	• `IN`: $n * m \times 1$ index vector of data points located within the polygon • `lon`: $c \times 1$ vector containing the longitude values of data points located within the polygon. • `lat`: $c \times 1$ vector containing the latitude values of data points located within the polygon.

function [param, dataApprox] = fitTrend2D (time, data, mode)	
Fit trend, annual and semi-annual signal	
Input	• `time`: $p \times 1$ vector of time in decimal years • `data`: $n \times m \times p$ data to be approximated by the estimated parameters, with n... number of longitudes, m... number of latitudes, p... number of time steps • `mode`: string, defining if the semiannual signal should be included in the estimation process (choose 'annual' or 'semiannual')
Output	• `param`: $n \times m \times \#param$ vector of estimated parameters (bias, trend, sine and cosine component of the annual signal, sine and cosine component of the semiannual signal) • `dataApprox`: $n \times m \times p$ data approximated by the estimated parameters

function [param, dataApprox] = fitIndex (time, data, index, mode)	
Fit trend, annual and semi-annual signal, and a selected climate index	
Input	• `time`: $p \times 1$ vector of time in decimal years • `data`: $n \times m \times p$ data to be approximated by the estimated parameters, with n... number of longitudes, m... number of latitudes, p... number of time steps • `index`: $p \times 1$ vector of climate index • `mode`: string, defining if the semiannual signal should be included in the estimation process (choose 'annual' or 'semiannual')
Output	• `param`: $n \times m \times \#param$ vector of estimated parameters (bias, trend, sine and cosine component of the annual signal, sine and cosine component of the semiannual signal, real-valued and imaginary component of the climate index) • `dataApprox`: $n \times m \times p$ data approximated by the estimated parameters

function [eigenvalues, E] = calculateEOF (Y)

Calculation of EOFs from a given data matrix $\mathbf{Y}$. The function calculates the p eigenvectors (=EOFs) corresponding to the p non-zero eigenvalues of the covariance matrix $\mathbf{C} = \mathbf{Y}\mathbf{Y}^T$. To reduce the computation effort, the eigenvalue problem is in a first step solved for the smaller matrix $\mathbf{C}' = \mathbf{Y}^T\mathbf{Y}$, which has the same eigenvalues. The eigenvectors of the larger matrix can then be calculated from the eigenvectors of the smaller matrix. The eigenvalues are stored in the vector `eigenvalues` sorted according to their magnitude, the eigenvectors are returned in the matrix $\mathbf{E}$.	
Input	• `Y`: matrix containing the time series of gridded data sets. The dimension is $n \times p$ with $n =$ number of grid points and $p =$ number of points in time.
Output	• `eigenvalues`: $p \times 1$ vector containing the p non-zero eigenvalues of the covariance matrix $\mathbf{C}$, sorted according to decreasing size. • `E`: $n \times p$ matrix containing in its columns the eigenvectors (EOFs) of the covariance matrix $\mathbf{C}$. EOFs are sorted according to size of corresponding eigenvalue.

function [D] = calculatePC (E, Y)	
Calculation of principle components (PCs) by projecting the original data onto the basis of the EOFs. The PCs are stored in the rows of the matrix $\mathbf{D}$.	
Input	• `Y`: matrix containing the time series of gridded data sets. The dimension is $n \times p$ with $n =$ number of grid points and $p =$ number of points in time. • `E`: $n \times p$ matrix containing in its columns the eigenvectors (EOFs) of the covariance matrix $\mathbf{C}$. EOFs are sorted according to size of corresponding eigenvalues.
Output	• `D`: $p \times p$ matrix containing the principle components in its rows.