Boring but important disclaimers:

- If you are not getting this from the GitHub repository or the associated Canvas page (e.g. CourseHero, Chegg etc.), you are probably getting the substandard version of these slides Don't pay money for those, because you can get the most updated version for free at

  `https://github.com/julianmak/academic-notes`

  The repository principally contains the compiled products rather than the source for size reasons.
- Associated Python code (as Jupyter notebooks mostly) will be held on the same repository. The source data however might be big, so I am going to be naughty and possibly just refer you to where you might get the data if that is the case (e.g. JRA-55 data). I know I should make properly reproducible binders etc., but I didn't...
- I do not claim the compiled products and/or code are completely mistake free (e.g. I know I don't write Pythonic code). Use the material however you like, but use it at your own risk.
- As said on the repository, I have tried to honestly use content that is self made, open source or explicitly open for fair use, and citations should be there. If however you are the copyright holder and you want the material taken down, please flag up the issue accordingly and I will happily try and swap out the relevant material.

OCES 3301 :

basic **Data Analysis** in ocean sciences

Session 10: fun with maps

Outline

(Just overview here; for actual content see Jupyter notebooks)

- actual fun with maps through `cartopy`

  → map projections

  examples with GEBCO and WOA13 data
- interpolation/extrapolation via an example
- Empirical Orthogonal Functions (EOFs)

  → basically PCA but with a spatial component (cf. 04 linear regression)

Digression: some geometry and topology

- location depends on choice of co-ordinates
- for the point $\boldsymbol{x} = (1, 1)$ in standard Cartesian co-ordinates, we could have

$$x = 1, \qquad y = 1$$

- but we could also have it in polar co-ordinates

$$r = \sqrt{2}, \qquad \theta = \pi/4$$

with $x = r\cos\theta$, $y = r\sin\theta$

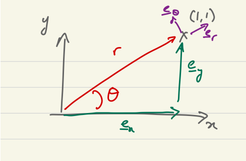


Figure: Polar co-ordinates schematic.

- some things more natural in polar co-ordinates
  $\rightarrow$ unit circle is the set of points with $r = 1$ and $\theta \in [0, 2\pi)$, compared to $y = \pm\sqrt{1 - x^2}$ for $x \in [-1, 1]$

Digression: some geometry and topology

- similarly, on a sphere, it is easier to consider spherical co-ordinates of (r, θ, ϕ) for radius, longitude and latitude

  $\rightarrow$ if on surface of Earth, take $r = R_{\mathrm{Earth}}$, so now 2d data

  $\rightarrow$ otherwise need a tuple (x, y, z) (raw satellite data do this actually)
- represent (lon, lat) data on 2d plane?

Digression: some geometry and topology

- similarly, on a sphere, it is easier to consider spherical co-ordinates of (r, θ, ϕ) for radius, longitude and latitude

  $\rightarrow$ if on surface of Earth, take $r = R_{\mathrm{Earth}}$, so now 2d data

  $\rightarrow$ otherwise need a tuple (x, y, z) (raw satellite data do this actually)
- represent (lon, lat) data on 2d plane?

!! problem: sphere is intrinsically 'curved' while a finite plane is intrinsically 'flat'

$\rightarrow$ unit sphere has Gaussian curvature $\kappa = 1$, while plane has $\kappa = 0$

$\rightarrow$ Gauss–Bonnet theorem tells you κ is related to the Euler characteristic χ (cf. think of χ as the sum of angles of a triangle on the surface)

geometry ('local', from κ) $\leftrightarrow$ **topology** ('global', from χ)

Digression: some geometry and topology

cursed cow

cursed doughnut

- topology studies global properties

  → e.g. how things are connected

  → cow is smoothly deformable into sphere

  → cup is smoothly deformable into doughnut

- cow is **not** smoothly deformable into doughnut

  → difference in genus (cf. 'holes')

Cartopy and map projections

- to go from sphere to plane, need to introduce a 'tear'

  (actually just removing one point will do it)

- main upshot: no global isometry between sphere and plane

  $\rightarrow$ i.e. **no distance preserving transformation possible**

Cartopy and map projections

- to go from sphere to plane, need to introduce a ‘tear’

  (actually just removing one point will do it)

- main upshot: no global isometry between sphere and plane

  $\rightarrow$ i.e. **no distance preserving transformation possible**

A map projection assigns (lon, lat) to some (x, y) on the plane via some formula, but implication from above is that we can at best, for a global map projection:

1. preserve angles (conformal map)
2. preserve areas (area-preserving map)
3. do neither

Cartopy and map projections

- to go from sphere to plane, need to introduce a ‘tear’

  (actually just removing one point will do it)

- main upshot: no global isometry between sphere and plane

  $\rightarrow$ i.e. **no distance preserving transformation possible**

A map projection assigns (lon, lat) to some (x, y) on the plane via some formula, but implication from above is that we can at best, for a global map projection:

1. preserve angles (conformal map)
2. preserve areas (area-preserving map)
3. do neither

`cartopy` package does most of the heavy lifting for you

Cartopy and map projections

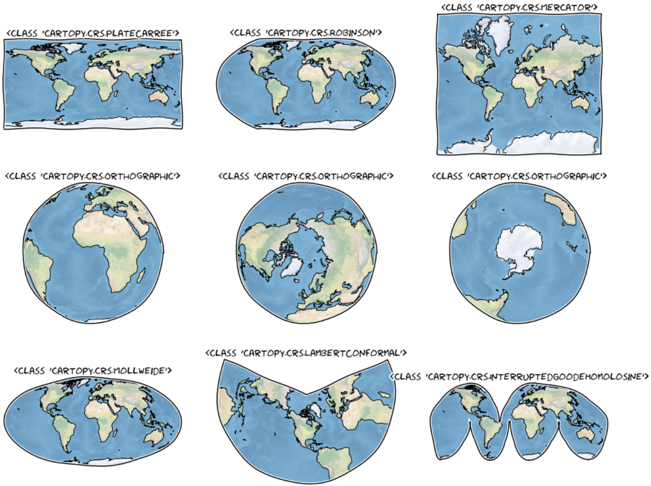


Figure: Sample projections available in `cartopy`. See notebook for `cartopy` usage and syntax.

Cartopy examples

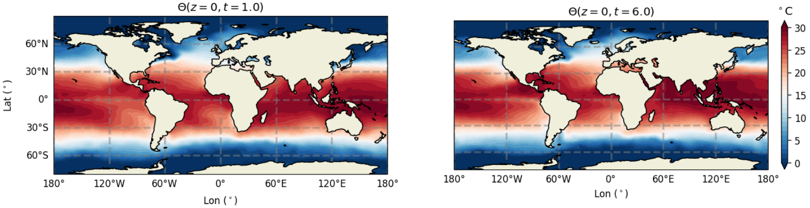


Figure: WOA13 data.

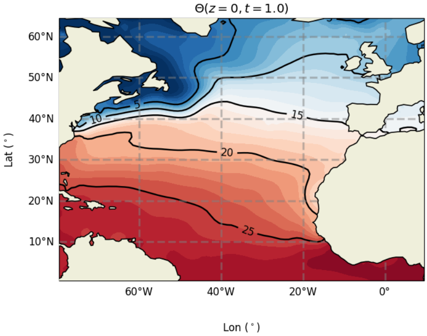
- WOA13 surface temperature from last time in Jan and Jul

  $\rightarrow$ xarray to read

  $\rightarrow$ cartopy Plate Carree projection

  $\rightarrow$ land features from cartopy

Cartopy examples



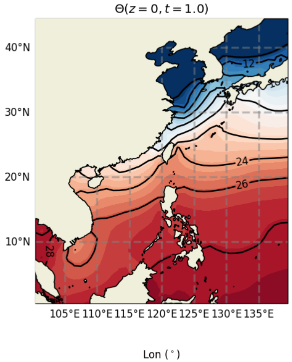


Figure: WOA13 data with zooms.

- WOA13 surface temperature centered over two different regions
  $\rightarrow$ xarray to read and subset
  $\rightarrow$ contours overlaid onto the colour plots

Cartopy examples

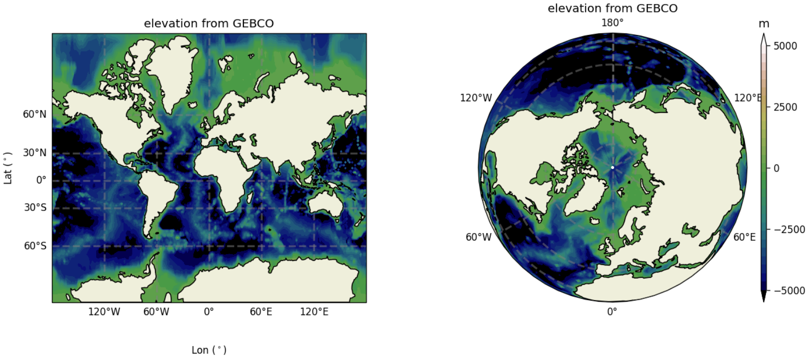


Figure: WOA13 data with zoms.

- GEBCO data using different projections
  → Mercator (conformal but really messes area up)
  → Northern Orthographic (does not show everything)

Interpolation/extrapolation (cf. 08)

- theory basically as before
  - → more caveats when more dimensions are involved
  - → mostly focus on the use, using `scipy.interpolate`
- demonstrative example from the paper below

Figure: ISME paper and steps to reproduce bits of Fig 1a.

Interpolation/extrapolation (cf. 08)

- want the SSS (sea surface salinity)

  $\rightarrow$ satellite products tend to be too coarse ($> 1/2^\circ$ horizontal resolution), and issues near coast

  $\rightarrow$ model output is finer, but other issues

- use the HYCOM dataset here

  $\rightarrow$ global calculation, data assimilated, around $1/12^\circ$ horizontal resolution

  $\rightarrow$ only need the greater bay area data

  $\rightarrow$ only need the month of June (grab all snapshots and time average)

Interpolation/extrapolation (cf. 08)

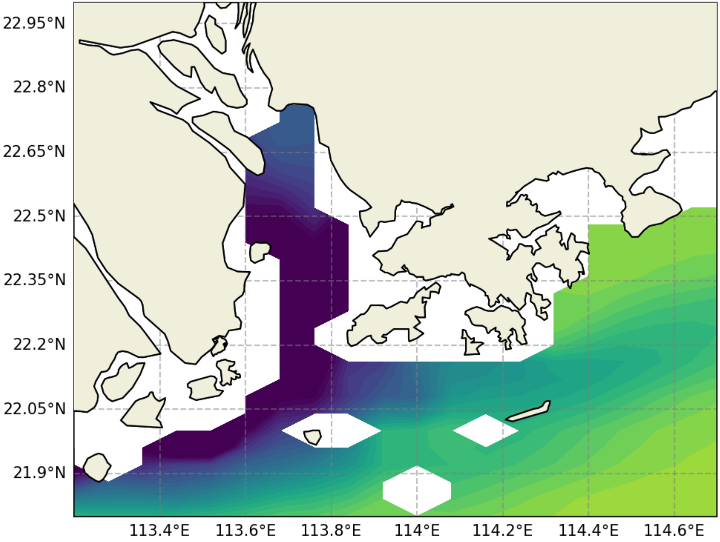
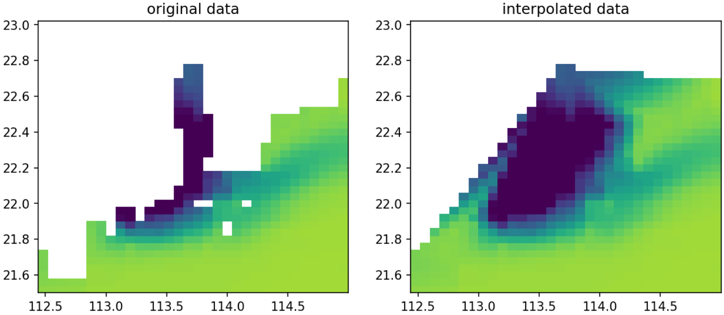


Figure: HYCOM SSS raw output time-averaged over June 2020, with Cartopy land features overlaid. Plot using `contourf`.

Interpolation/extrapolation (cf. 08)

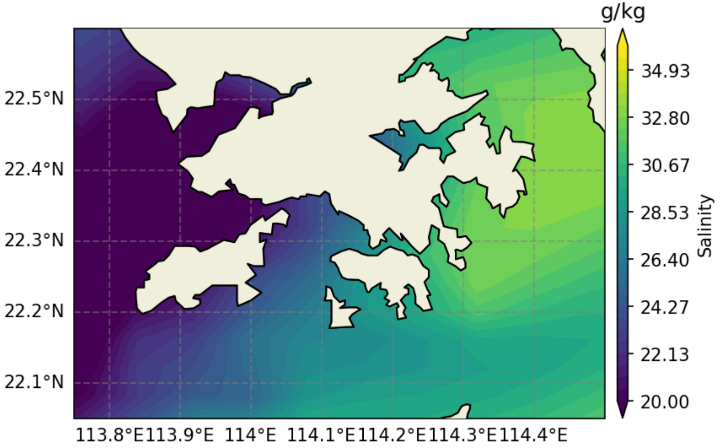


(left) raw plot as pixels (`pcolors`), masked places are land points

(right) interpolated data (used `CloughTocher2DInterpolator`)

Q. only some points are filled, what is the condition for a valid inteprolation?

Interpolation/extrapolation (cf. 08)



- pass to caropy, overlay land features, format accordingly
  → probably fine here as a visual schematic

EOFs

Want to do something like

$$f(t,x,y) = \sum_{k=1}^{N} \mathrm{PC}_k(t)\,\mathrm{EOF}_k(x,y)$$

to pick out spatial patterns that capture the most variability

- sound familiar?

EOFs

Want to do something like

$$f(t,x,y) = \sum_{k=1}^{N} \mathrm{PC}_k(t)\, \mathrm{EOF}_k(x,y)$$

to pick out spatial patterns that capture the most variability

- sound familiar? basically like PCA! (04 regression)

  → $\mathrm{EOF}_k(x,y)$ the Empirical Orthogonal Function (EOF)

  → EOF tagged with a Principal Component $\mathrm{PC}_k(t)$
- algorithm and methodology largely the same

  → actually going to use the Singular Value Decomposition (SVD cf. diagonalisation of covariance matrix for PCA)

  → going to leverage `scikit-learn` package again

- EOF analysis finds you a spatial basis via data (one could choose it in advance also; cf. Fourier analysis)

EOFs: work flow

- start with array containing $f(t, x, y)$, flatten into $f(t, \text{space})$
  $\rightarrow$ "space" is now the categorisation, and t contains the data points
  $\rightarrow$ cf. Iris sepal length vs. entries of Iris sepal length etc.

- preprocessing, but a few choices:
  $\rightarrow$ de-mean, de-trend, Z-score standard scaling, others...

- throw into PCA algorithm (or your SVD algorithm if you want)
  $\rightarrow$ unflatten the resulting EOF(space) back to EOF(x, y)
  $\rightarrow$ EOFs ranked in variance explained

EOFs: idealised example

Try first for (cf. data made for animation)

$$f(x, y) = \sin(x)\cos(y)\sin(t) + \frac{1}{2}\cos(2x)\cos(2t)$$

- 1st term as is, circular blobs
- 2nd term are ‘rolls’
- power spectrum would definitively pick out two peaks

  (which ones? ignoring isolated values of t where individual parts of the function vanish)

Q. what would the EOF analysis do?

EOFs: idealised example

Try first for (cf. data made for animation)

$$f(x, y) = \sin(x)\cos(y)\sin(t) + \frac{1}{2}\cos(2x)\cos(2t)$$

- 1st term as is, circular blobs
- 2nd term are ‘rolls’
- power spectrum would definitively pick out two peaks

  (which ones? ignoring isolated values of t where individual parts of the function vanish)

Q. what would the EOF analysis do?

$\rightarrow$ EOF_1 to be the 1st term, with $\mathrm{PC}_1 \sim \sin(t)$?

(because of imposed larger amplitude)

$\rightarrow$ EOF_2 to be the 2nd term, with $\mathrm{PC}_2 \sim \cos(2t)$?

EOFs: idealised example, no preprocessing

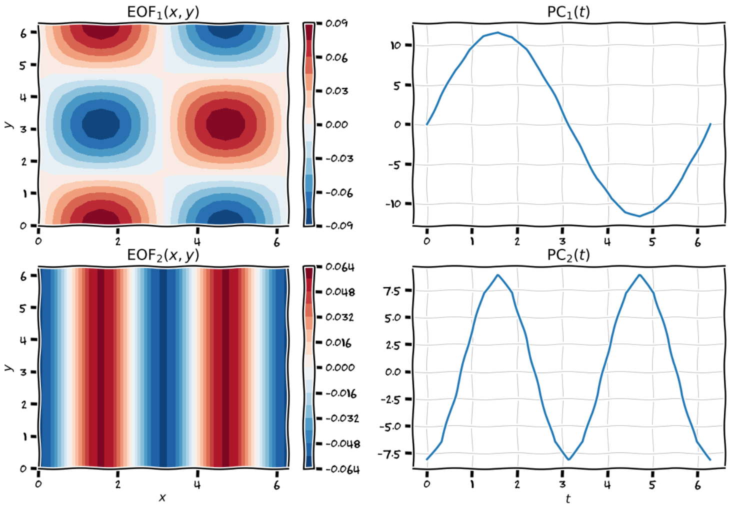


Figure: EOF of idealised data, no preprocessing.

EOFs: idealised example, demean

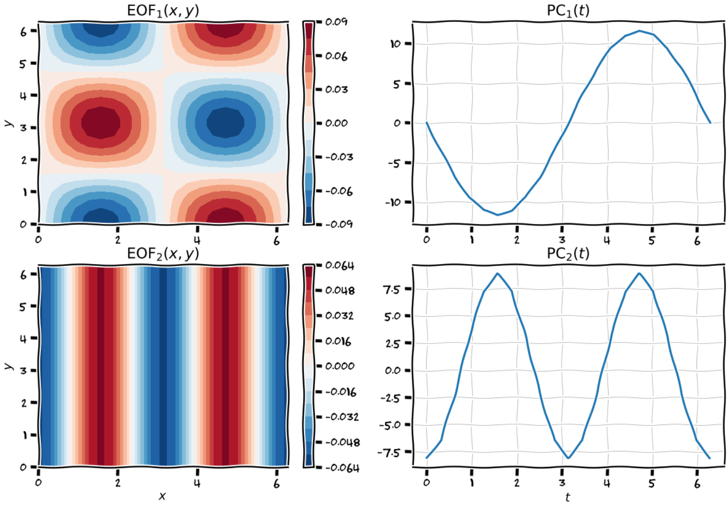


Figure: EOF of idealised data, remove time average per point. Notice a sign flip of EOF and PC 1, but that's ok.

EOFs: idealised example, detrend

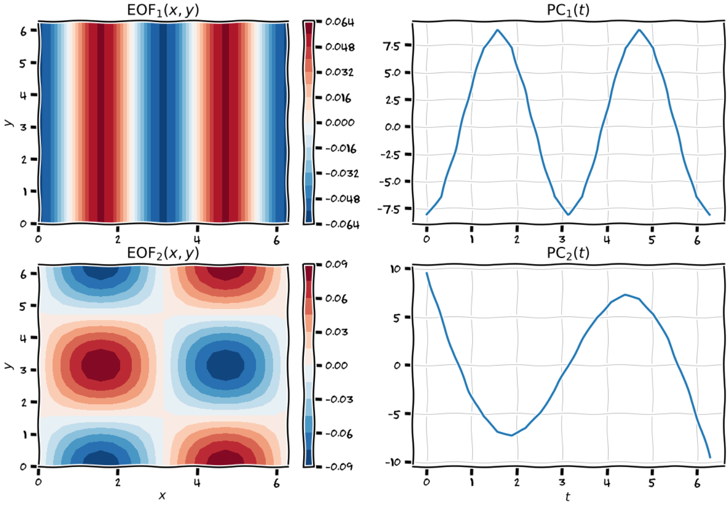


Figure: EOF of idealised data, remove linear trend per point (`scipy.signal.detrend` or use `numpy.polyfit`). Notice 'swapping' of EOF ordering relative to previous case, and PC 2 looks a bit weird.

EOFs: idealised example, Z-score standardisation

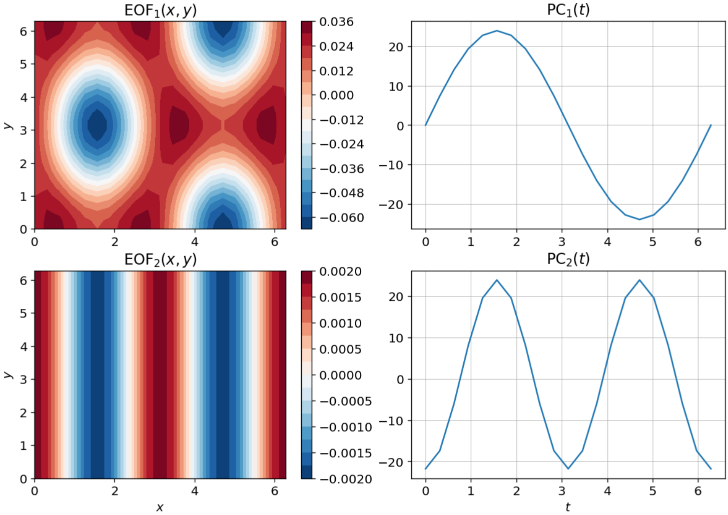


Figure: EOF of idealised data, using Z-score standardisation (`StandardScaler` in `scikit-learn`). Note that while EOF 1 no longer looks like circular blobs, and EOF 2 sign has changed, these are still valid choices of basis.

EOFs: 'real' example

- provided Extended Reconstructed SST data (full and anomaly version)

  → monthly, from mid 1800s to present day

  → global, 2° spatial resolution

  → already masked

  → anomaly relative to some climatology (!?)

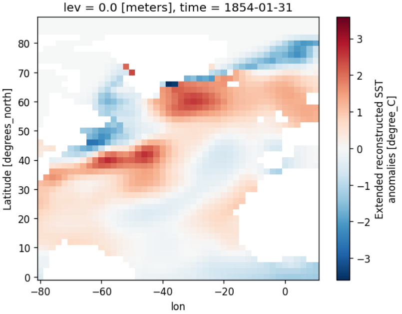


Figure: Raw data plot of SST **anomalies** over Atlantic. Using the anomalies file directly.

EOFs: ‘real’ example

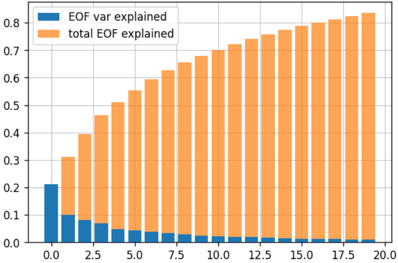


Figure: Percentage and cumulative percentage of variance explained by EOFs.

- EOF analysis as usual

  $\rightarrow$ using anomalies file directly, not detrending or demeaning here
- variance explained of EOFs are generally pretty low actually...

EOFs: ‘real’ example

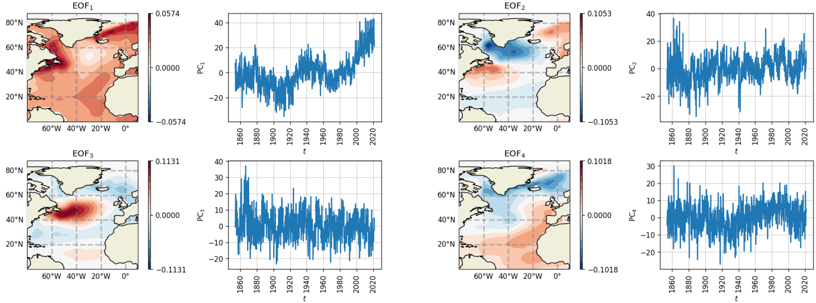


Figure: EOFs. Probably (EOF1) global warming with some Atlantic Multi-decadal Variability (AMV), (EOF2) North Atlantic Oscillation (NAO), (EOF3) looks like EOF 2 of Fig. 3 in Buckley et al. 2013, and (EOF4) looks a bit like the AMV?

Jupyter notebook

go to 10 Jupyter notebook to get some code practise

- exercises and code I haven't demonstrated here

  $\rightarrow$ EOFs with different locations and/or different pre-procesing

  $\rightarrow$ other choices of interpolators

  $\rightarrow$ analysis of PCs computing its power spectrum

  $\rightarrow$ …