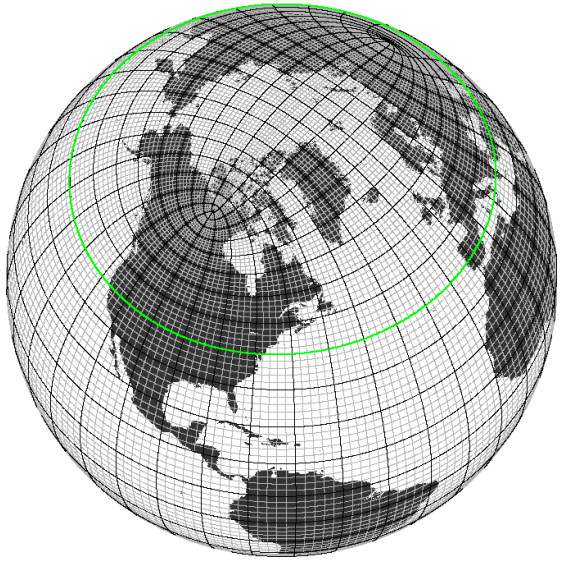
Lecture 3
Gridded data and interpolation methods

Gridded Data

A few examples of numerical grids:

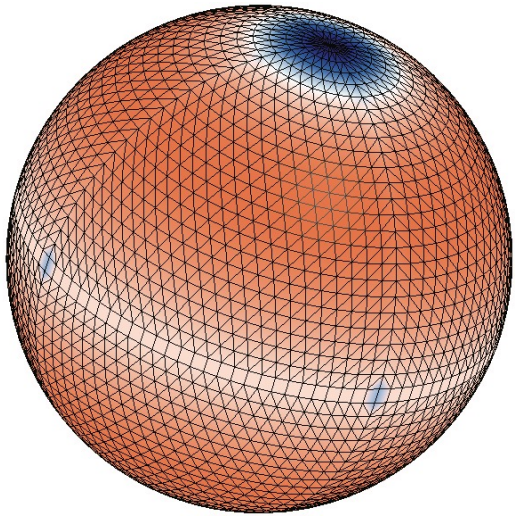
FINITE DIFFERENCE MODELS

Latitude-Longitude Grid
(currently used by e.g. UK Met Office)



Tripolar Grid
(used by all ORCA -family ocean and sea ice models, e.g. NEMO, CICE)

SPECTRAL MODEL



Cubic-octahedral Grid
(used by ECMWF)

Gridded Data

Prognostic equations must be discretized in time (see Numerical Methods) and SPACE.

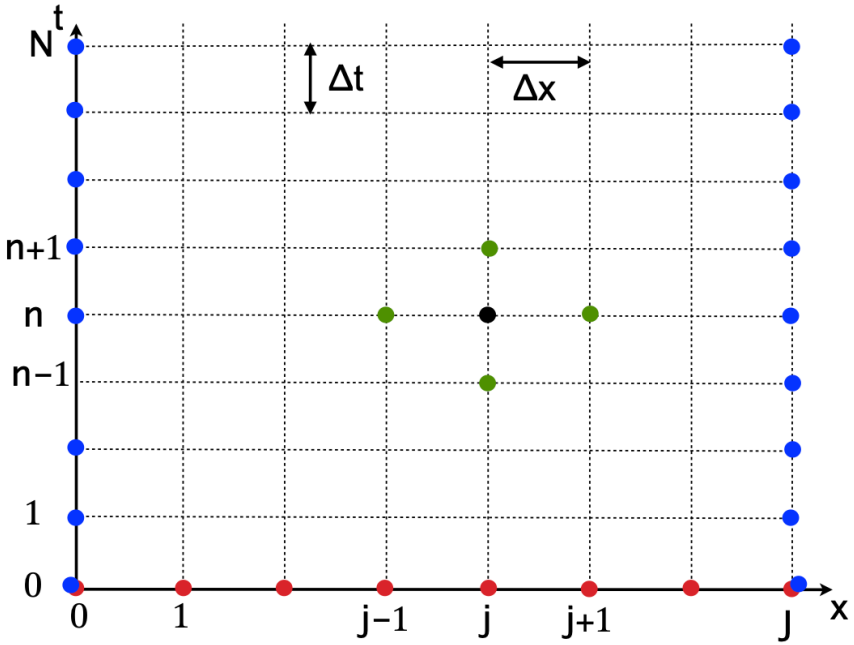


Figure 4.1 of Doos K. et al., Stockholm University Press

Gridded Data

Similarly to the time discretization, discretizing the equations in space also introduces an error: added numerical diffusion that damps the numerical solution.

Shallow Water (SW) equation (gravity waves)

$$\frac{\partial h}{\partial t} + H\frac{\partial u}{\partial x} = 0$$

$$\frac{\partial u}{\partial t} + g\frac{\partial h}{\partial x} = 0$$

Wave phase speed: $c = \frac{\omega}{k} = \pm\sqrt{gH}$

g: $gravity\ acceleration, H$: $height/depth$

Forward in time Centered in Space

$$h_j^{n+1} = h_j^n - H\frac{\Delta t}{2\Delta x}\left(u_{j+1}^n - u_{j-1}^n\right)$$

$$u_j^{n+1} = u_j^n - g\frac{\Delta t}{2\Delta x}\left(h_{j+1}^n - h_{j-1}^n\right)$$

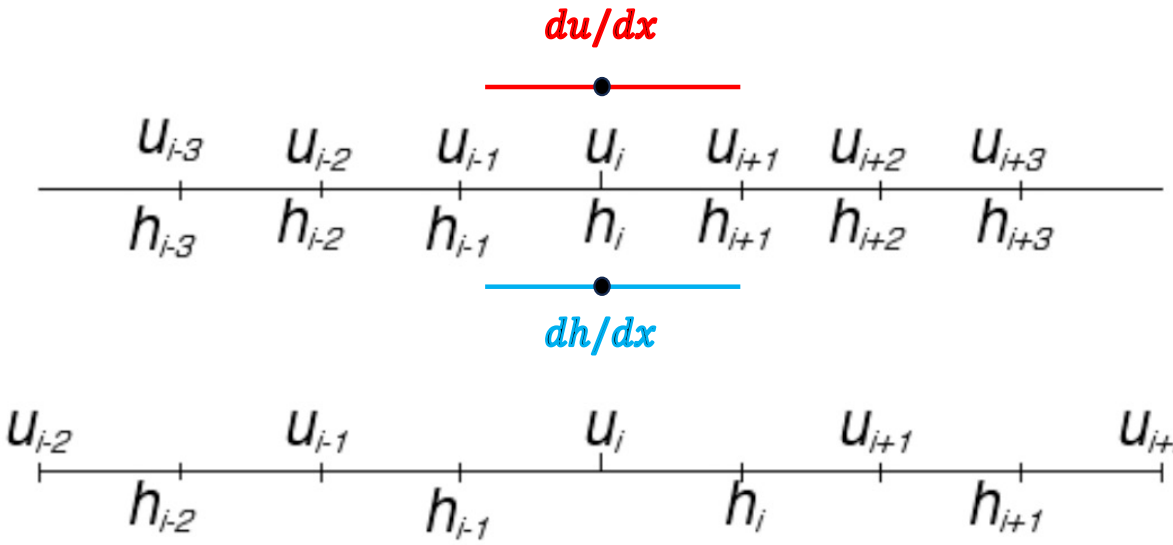
Group phase speed of the numerical solution:

$$C_{Dg} = \frac{d\left(kC_D\right)}{dk} = c\cos\left(k\Delta x\right)$$

DISPERSIVE

Gridded Data

By choosing the discretization method (i.e. the type of grid) one can minimize this error:



UNSTAGGERED GRID (A grid)

STAGGERED GRID (B,C,D,E grids)

Figure 8.1 of Doos K. et al., Stockholm University Press

Gridded Data

Arakawa's A grid for 3D SW equation:

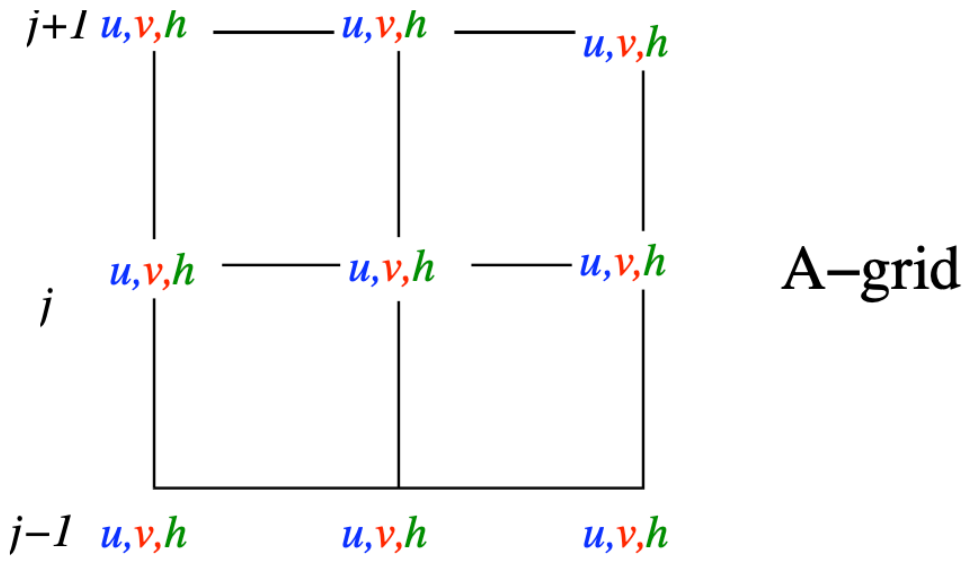


Figure 8.3 of Doos K. et al., Stockholm University Press

Solution is unstable and highly dispersive.

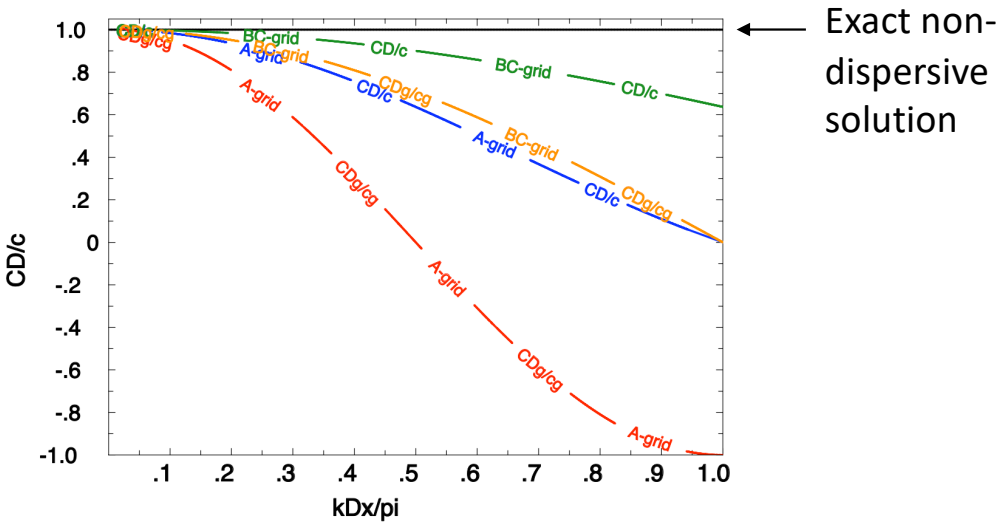


Figure 8.2 of Doos K. et al., Stockholm University Press.
Numerical wave phase speed for A,B and C grid.

Gridded Data

Arakawa's B and C grids for 3D SW equation:

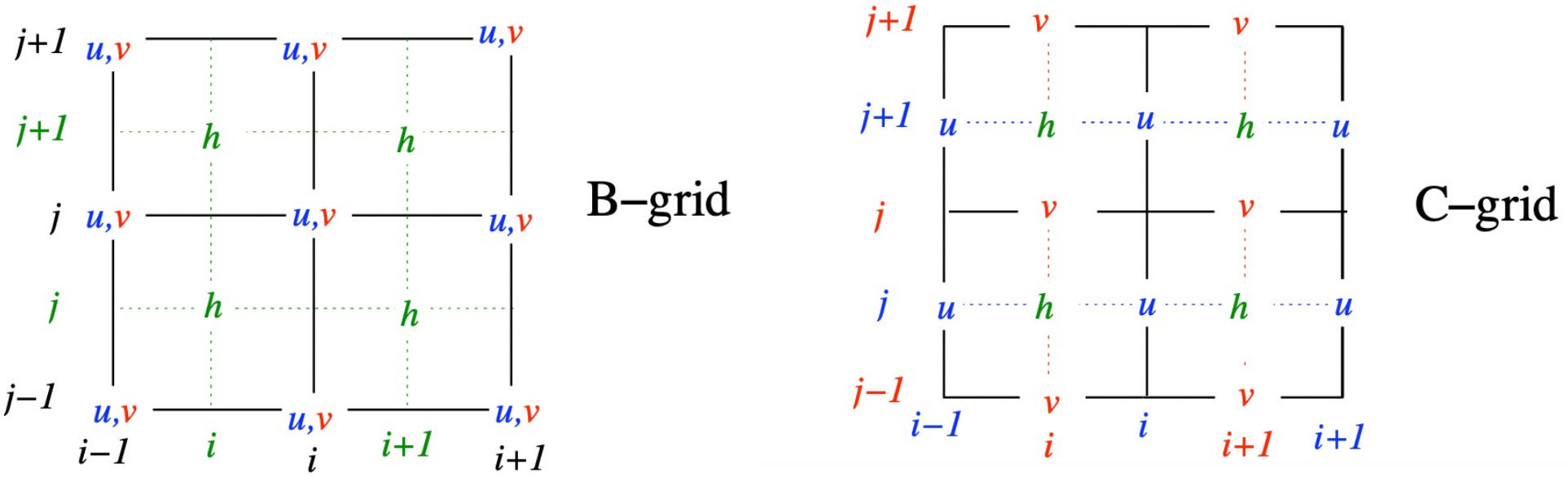


Figure 8.3 of Doos K. et al., Stockholm University Press

In the C-grid the surface elevation h is calculated at the centre of each grid box, while the velocities u and v are calculated at the boundaries. In this way, gradients of h are calculated where u and v are located and gradients of u and v are calculated where h is located.

Gridded Data

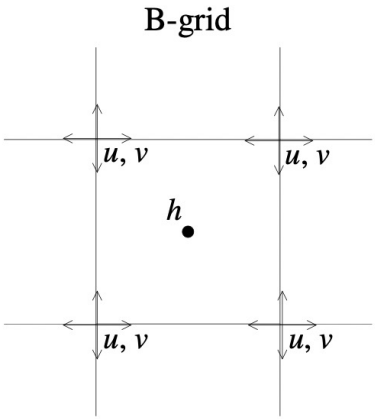
Most models use the C grid staggering to discretize and solve the prognostic equations, with vector quantities (wind, stress, etc..) defined along the grid-cell interfaces and all others quantities (mass, temperature, pressure, etc.) in the centre of the grid-cell.

Note that working on a staggered grid means, effectively, working with a number of different grids:

- T grid
- U grid
- V grid

What does it mean for you?

```
>>> print (cubes)
0: air_pressure_at_sea_level / (Pa)    (time: 780; latitude: 144; longitude: 192)
...
3: x_wind / (m s-1)                    (time: 780; pressure: 17; latitude: 145; longitude: 192)
4: y_wind / (m s-1)                    (time: 780; pressure: 17; latitude: 145; longitude: 192)
```



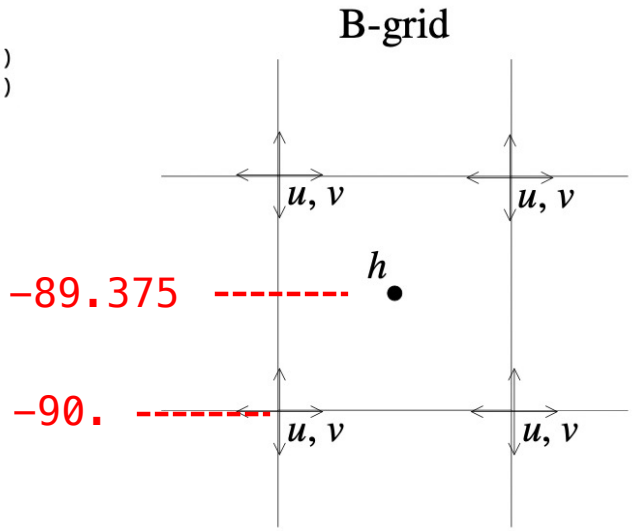
Gridded Data

OUTPUT from UKESM (UK Earth System Model)

```
>>> print (cubes)
0: air_pressure_at_sea_level / (Pa)    (time: 780; latitude: 144; longitude: 192)
…
3: x_wind / (m s-1)                    (time: 780; pressure: 17; latitude: 145; longitude: 192)
4: y_wind / (m s-1)                    (time: 780; pressure: 17; latitude: 145; longitude: 192)
```

```
print (cubes[0].coords('latitude'))
[DimCoord(array([-89.375, -88.125, -86.875, -
85.625, -84.375, -83.125, -81.875,… ]
```

```
print (cubes[3].coords('latitude'))
[DimCoord(array([-90.  , -88.75, -87.5 , -86.25,
-85.  , -83.75,… ]
```



OFFSET between variables on the grid: this can become an ‘issue’ when you perform math operations on the gird. SOLUTION: re-gridding!

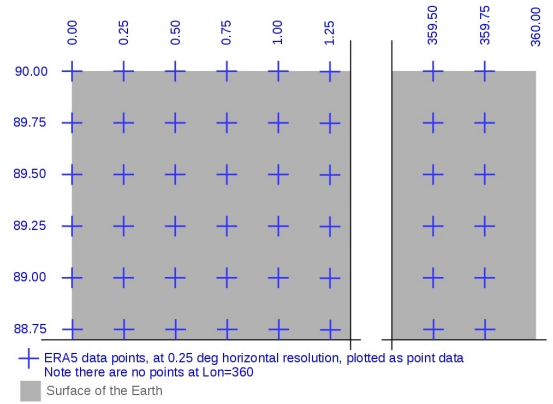
Gridded Data

OUTPUT from ECMWF model (ERA5 reanalysis)

```
>>> import iris
>>> cube_U=iris.load_cube('ua_1950_01.nc')
>>> cube_V=iris.load_cube('va_1950_01.nc')
>>> cube_T=iris.load_cube('tas_1950_01.nc')
>>> print (cube_U)
eastward_wind / (m s**-1)           (time: 1; pressure_level: 37; latitude: 721; longitude: 1440)
    Dimension coordinates:
        time                            x                 -             -              -
        pressure_level                  -                 x             -              -
        latitude                        -                 -             x              -
        longitude                       -                 -             -              x
    Attributes:
        Conventions                 'CF-1.6'
        history                     '2023-06-15 10:57:56 GMT by grib_to_netcdf-2.25.1: /opt/ecmwf/m
>>> print (cube_V)
northward_wind / (m s**-1)          (time: 1; pressure_level: 37; latitude: 721; longitude: 1440)
    Dimension coordinates:
        time                            x                 -             -              -
        pressure_level                  -                 x             -              -
        latitude                        -                 -             x              -
        longitude                       -                 -             -              x
    Attributes:
        Conventions                 'CF-1.6'
        history                     '2023-06-15 10:58:08 GMT by grib_to_netcdf-2.25.1: /opt/ecmwf/m
>>> print (cube_T)
2 metre temperature / (K)           (time: 1; latitude: 721; longitude: 1440)
    Dimension coordinates:
        time                            x             -              -
        latitude                        -             x              -
        longitude                       -             -              x
    Attributes:
        Conventions                 'CF-1.6'
        history                     '2024-02-05 14:43:47 GMT by grib_to_netcdf-2.25.1: /opt/ecmwf/m
```

IFS is a Spectral Model!

The time discretization is not done used finite differences. During post-processing the model outputs are interpolated on a regular lat-lon grid:



https://confluence.ecmwf.int/display/CKB/ERA5%3A+What+is+the+spatial+reference

Interpolation Methods

Common situations in which you will need to interpolate (or regrid) your data:

-datasets have a different grid-spacing: e.g., climate model output (~100km) and ERA5 reanalysis (~ 30 km);
-need to move from one grid type to another: e.g., from tripolar to regular latitude-longitude grid;
-need to compare station data with model outputs: how to find variables at the same exact location?
-variables have been computed on different grids: e.g., UV grid *vs* T grid.

`Python IRIS` interpolation/regridding schemes for gridded data :

iris.analysis.Linear
iris.analysis.Nearest
iris.analysis.PointInCell
iris.analysis.AreaWeighted

LINEAR and NEAREST NEIGHBOR interpolation methods

https://scitools-iris.readthedocs.io/en/latest/userguide/interpolation_and_regridding.html

Interpolation Methods

Alternative generic python function:

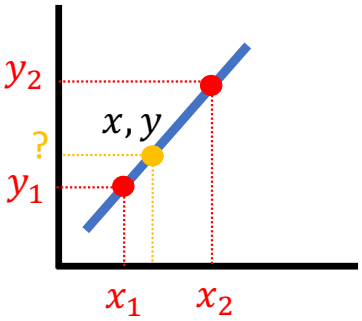
`scipy.interpolate.griddata()`

https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.griddata.html

Interpolating vs Regridding: very similar processes but interpolation is based on single points, while regridding is based on the horizontal grid of a second datasets.

LINEAR INTERPOLATION

Formula for the slope: $\overbrace{\Delta y/\Delta x}$

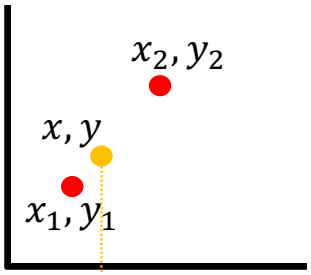


$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y}{x_2 - x}$$

$$y = y_1 + \frac{y_2 - y}{x_2 - x}(\mathrm{x} - x_1)$$

NEAREST NEIGHBOR INTERPOLATION

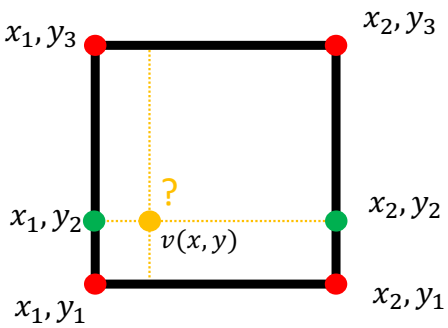
It finds the closest point:



$$y = y_1$$

Interpolation Methods

BILINEAR INTERPOLATION

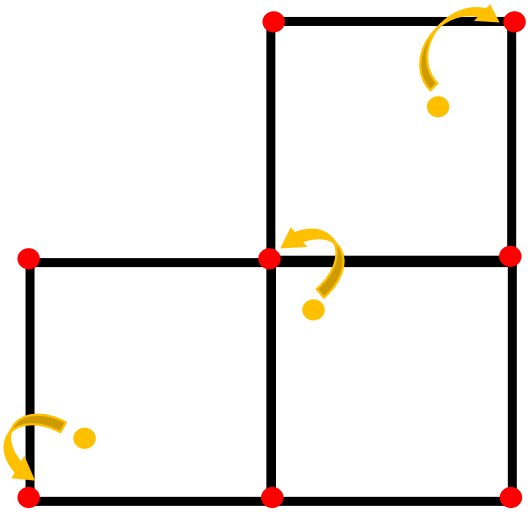


Linear interpolation to find $v(x_1, y_2)$

Linear interpolation to find $v(x_2, y_2)$

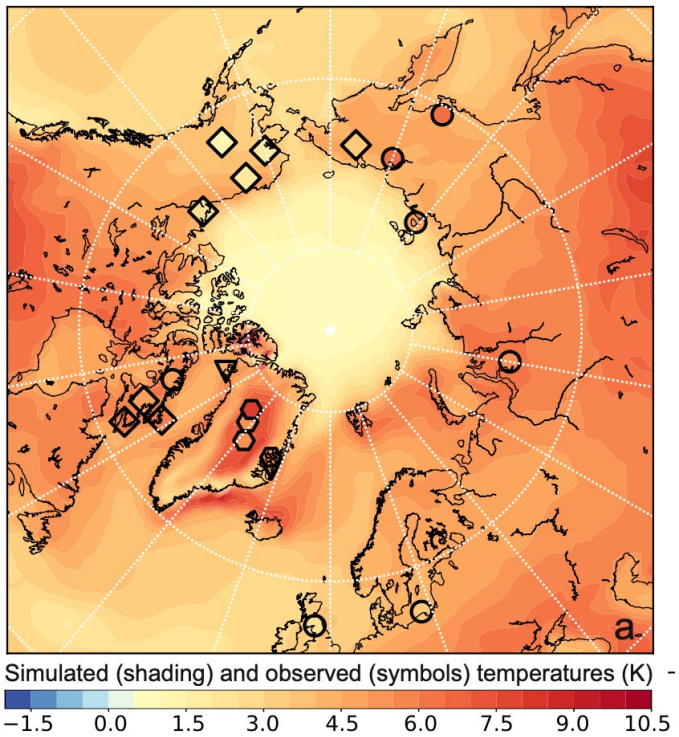
Linear interpolation to find $v(x \quad , y \quad)$
using $v(x_1, y_2)$ and $v(x_2, y_2)$

2D NEAREST NEIGHBOR INTERPOLATION



Interpolation Methods

Comparing station data with model outputs: INTERPOLATION



From Guarino et al., 2020, NCC.

Lat, Lon	Site	Observations (K)	HadGEM3 (K)
Terrestrial			
55, 18	Europe	3.4 ± 0.5*	4.3 ± 1.2
55, -3	UK	2.4 ± 0.5*	3.5 ± 1.0
61, 152.5	Magadan	6.4 ± 2	4.5 ± 1.4
68, 80	West-central Siberia	5.4 ± 2	6.4 ± 2.1
68, 160	Northeast Siberia	6.4 ± 2	4.4 ± 2.0
70, -72.5	Flitaway	4.9 ± 0.5	5.8 ± 1.7

Interpolation Methods

Comparing station data with model outputs: INTERPOLATION

```
import iris
import numpy as np

## Load Surface Air Temperature dataset ##
SAT=iris.load_cube('/path_to_data/uar766_SAT15m_18502050_monthly.nc',  'air_temperature')


## Extract SAT value at exact model grid-box and write out in a file ##
data_in=np.loadtxt('/home/path_to_data/observations.txt', skiprows=1)
lat, lon = data_in[:,0], data_in[:,1] # Read in location of obs sites in degrees
f=open('HadGEM3_sites', 'w+')

for i in range(0,21): #21=number of sites
 site= [('latitude', lat[i] ), ('longitude',  lon[i])]
 #HadGEM3=SAT.interpolate(site, iris.analysis.Linear())
 HadGEM3=SAT.interpolate(site, iris.analysis.Nearest())

 print lat[i], lon[i], HadGEM3.data
 f.write(str(lat[i])+'   ' + str(lon[i])+'  ' +str(HadGEM3.data))
```

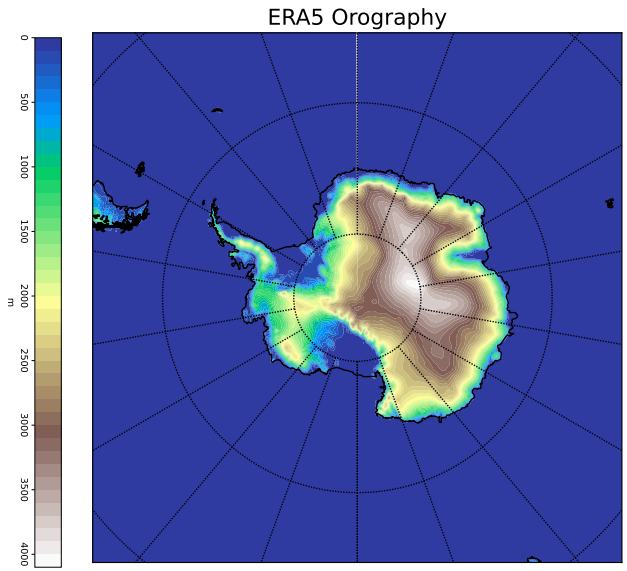
Interpolation Methods

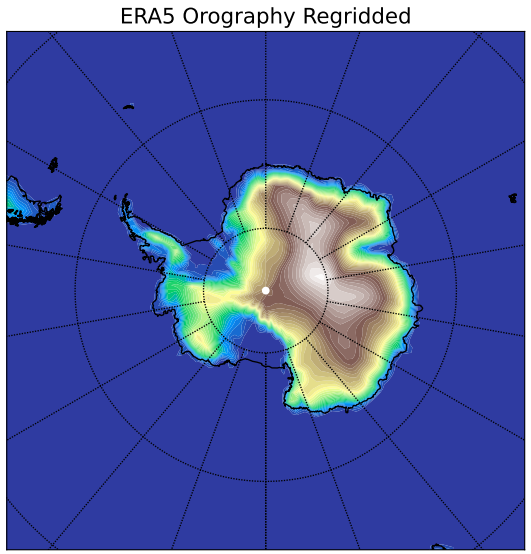
From high-resolution to coarse resolution: REGRIDDING

```
def regrid(data_in, grid, binary):

 if binary:
  data_coarse=data_in.regrid(grid, iris.analysis.Nearest())
 else:
  data_coarse=data_in.regrid(grid, iris.analysis.Linear())
  #print (data_in, data_in_coarse)


 #let's check the regridding result
 '''
 plt.figure()
 qplt.contourf(data_in[0], 25)
 plt.figure()
 qplt.contourf(data_coarse[0], 25)
 plt.show()
 '''


 return(data_coarse)
```

Interpolation Methods

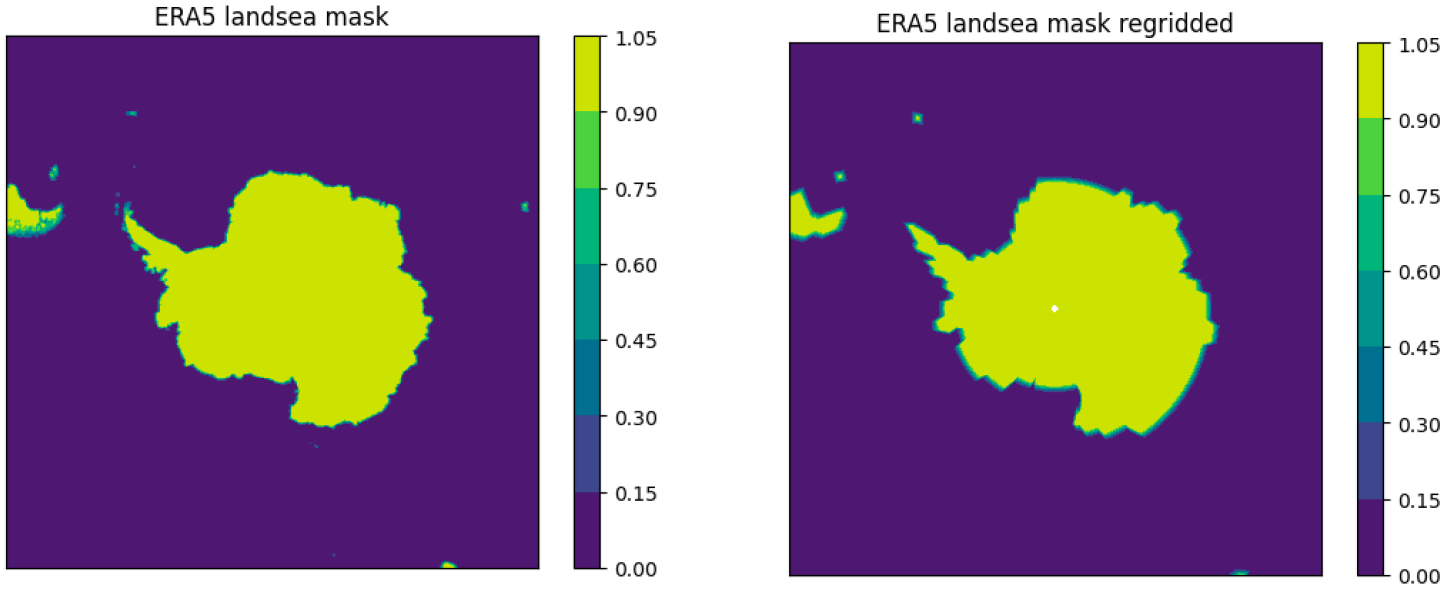
```
#Load target grid
target_grid=iris.load_cube('/home/clima-archive5/mguarino/UKESM_grid.nc')
#Load ERA5 orograpy
H=iris.load_cube('/home/clima-archive5/ERA5/fixed/orography_m.nc')
print (H)
#Call function to regrid the high-res ERA5 data into the coarser resolution model grid
H=regrid(H, target_grid, binary=False)
```





Interpolation Methods

```
#Load target grid
target_grid=iris.load_cube('/home/clima-archive5/mguarino/UKESM_grid.nc')
#Load ERA5 landsea mask
LS_mask=iris.load_cube('/home/clima-archive5/ERA5/fixed/landmask.nc')
print (LS_mask)
#Regrid the high-res ERA5 data into the coarser resolution model grid
LS_mask=regrid(LS_mask, target_grid, binary=True)
```



Exercises

Use the provided 2D gridded datasets and:

1. Re-grid the coarse resolution (model) dataset into the high-resolution (observations) one.
   What re-gridding scheme are you using, and why?

2. Produce a map of your two input datasets and of the re-gridded one (see example provided). To do this, write a 'plotting function' that you can call anytime you need to plot your data.

3. Now that model and observations are on the same grid, compute the difference between the two and plot the resulting field.

4. Using the anomaly field, extract the grid-point values at the following locations:
LATITUDE , LONGITUDE
41.9, 12.5
37.8, 122.4
-16.5, 68.12
-33.86, 151.20

Exercises

Example script: **DA_exercises_3.py**

Mean Sea Level Pressure from ERA5 reanalysis, long term mean from 1950 to 2014:
`MSLP_ERA5_1950_2014_mean_30km.nc`

Mean Sea Level Pressure from UKESM model run, long term mean from 1950 to 2014:
`MSLP_UKESM_1950_2014_mean_100km.nc`

```
#Load high resolution dataset  (ERA5 reanalysis)
D1=iris.load_cube('MSLP_ERA5_1950_2014_mean_30km.nc')
print (D1)
#Load lower resolution dataset (UKESM model outputs)
D2=iris.load_cube('MSLP_UKESM_1950_2014_mean_100km.nc')
print (D2)

#let's have a look at the latitude coordinate
print (D1.coords('latitude'))
print (D2.coords('latitude'))
```

Exercises


```
print (D1)
```

```
mslp / (hPa)                          (latitude: 721; longitude: 1440)
    Dimension coordinates:
        latitude                                x                   -
        longitude                               -                   x
    Scalar coordinates:
        time                                  1982-06-17 00:00:00,
                                              bound=(1950-01-01 00:00:00, 2014-12-01 00:00:00)
    Cell methods:
        0                                     time: mean
```

```
print (D2)
```

```
mslp / (hPa)                          (latitude: 144; longitude: 192)
    Dimension coordinates:
        latitude                                x                   -
        longitude                               -                   x
    Scalar coordinates:
        forecast_reference_time               1850-01-01 00:00:00
        time                                  1982-07-01 00:00:00,
                                               bound=(1950-01-01 00:00:00, 2015-01-01 00:00:00)
```

Exercises


```
from mpl_toolkits.basemap import Basemap


#Example of how to produce a map using Basemap

fig=plt.figure()
ax=plt.gca()

bmap=Basemap(projection= 'gall', llcrnrlat= -90,  urcrnrlat= 90,
llcrnrlon=0,  urcrnrlon= 360, resolution='l')
lon= D1.coord('longitude').points
lat= D1.coord('latitude').points
x,y=bmap(*np.meshgrid(lon,lat))
contours=bmap.contourf(x,y, D1.data, levels=20, cmap='jet')

bmap.drawcoastlines()
plt.colorbar()
plt.title(' ')

plt.show()
```