Python scripts to help users explore Baltic+ SEAL products

By Emma Chalençon (UCC) & Marcello Passaro (TUM)

Baltic+ SEAL Project

See www.balticseal.eu *for more information*

1. Plotting Baltic+ SEAL products

1.1 How to plot a Baltic+ SEAL along-track product?

This code is a sample plot for users to plot Baltic+ SEAL along-track altimetry sea level measurements

These codes were produced using **Python 3** (Version 3.8); The following packages have to be installed on your python: **numpy**, **scipy**, **netCDF4**, **matplotlib** and **cartopy**. You can install the packages using pip (for example "`pip install numpy`") or conda (for example "`conda install numpy`").

Note: Cartopy installation may be a bit complex as it has a lot of required dependencies. Using pre-built binaries can be a solution. They can be found at a variety of sources. Christoph Gohlke maintains unofficial Windows binaries (https://www.lfd.uci.edu/~gohlke/pythonlibs/). The correct version of the package of interest (ex: cp38 and win_amd64 is for Python 3.8 (64bits)) can be downloaded and installed by using "`pip install`*" followed by the path of the .whl file which has just been downloaded (*`pip install C:/some-dir/some-file.whl`*).*

```
# Importing the needed packages:

import netCDF4
import numpy as np
import cartopy as cart
from cartopy.mpl.ticker import LongitudeFormatter, LatitudeFormatter
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
from scipy.interpolate import griddata

# Setting the inputs:

directory='C:/some-dir/' #The user should change to the .nc files' directory
filename='file.nc' #The user should change to the right file
S = netCDF4.Dataset(directory+filename)
lon = S.variables['lon'][:]
lat = S.variables['lat'][:]
ssh = S.variables['ssh'][:]
min_lat=53.0
max_lat=66.0
min_lon=9.0
max_lon=31.0


# Displaying the along-track:
fig = plt.plot()
plt.rcParams.update({'font.size': 15})
plt.plot
plt.rcParams["figure.figsize"] = (50,10) #Increase figure size
ax = plt.axes(projection=ccrs.Miller())
img=plt.scatter(lon, lat,
         c=ssh, s=20,
         cmap='cool', alpha=1,transform=ccrs.PlateCarree())
ax.coastlines(resolution='10m', color='black', linewidth=1)
ax.set_xticks(np.arange(min_lon,max_lon,2), crs=ccrs.PlateCarree())
ax.set_yticks(np.arange(min_lat,max_lat,1), crs=ccrs.PlateCarree())
lon_formatter = cart.mpl.ticker.LongitudeFormatter(number_format='.1f',
                                   degree_symbol='',
                                   dateline_direction_label=True)
lat_formatter = cart.mpl.ticker.LatitudeFormatter(number_format='.1f',
                                  degree_symbol='')
ax.xaxis.set_major_formatter(lon_formatter)
ax.yaxis.set_major_formatter(lat_formatter)
plt.colorbar(img,label=r'SSH (m)')
plt.clim(15, 40)
plt.show() #A window will show up, allowing the user to see and download the plot
```

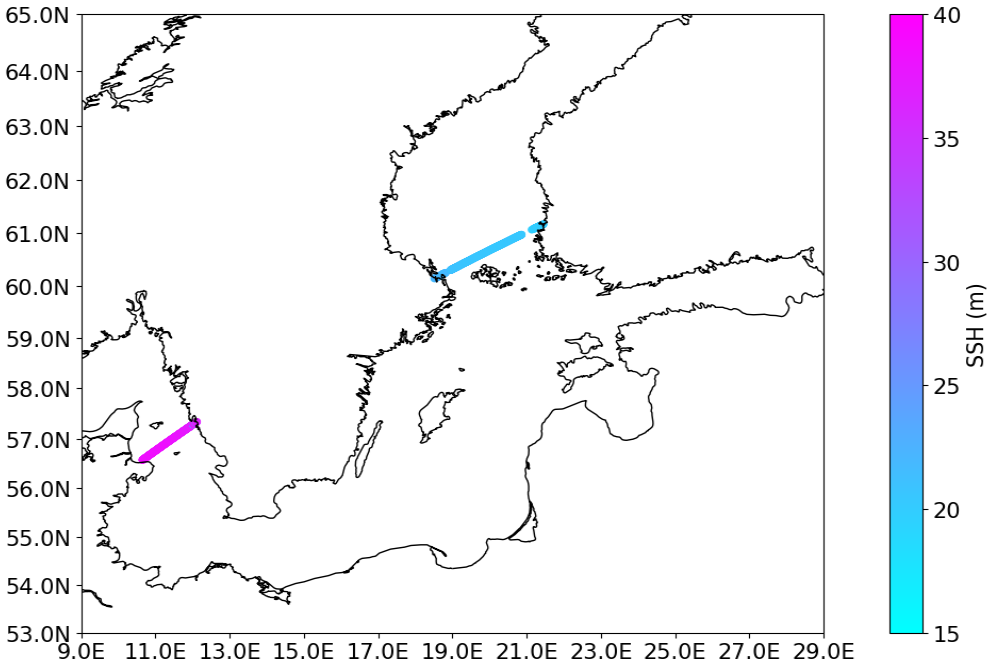

Figure 1: *Output of the python code: Baltic SEAL along-track data (jason1_em_hf_262_0213.nc)*

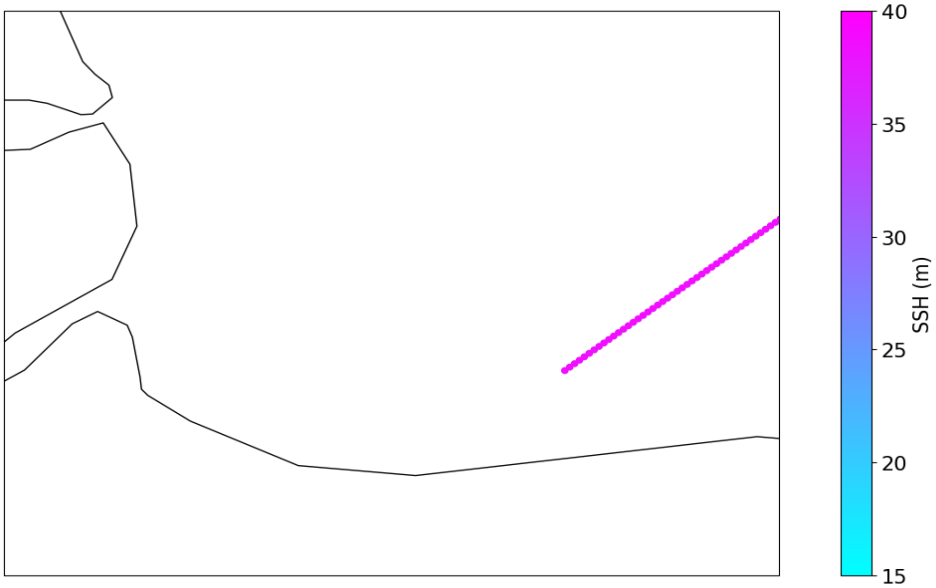

Figure 2: *Output of the python code: Zoom on Treå Møllebugt (Denmark)*

1.2 How to plot a Baltic+ SEAL gridded product?

This code is a sample plot for users to plot a grid of Baltic SEAL

It is important to note that Baltic SEAL is distributed using unstructured grids. Therefore, the points of the original grid shall be displayed in a "scatter plot" (see the 1.2.1 python code). Since several ocean data are distributed onto structured grids, it might be useful to interpolate a Baltic SEAL product on such a grid. The 1.2.2 code provides an example by defining a grid with a 1/10th-of-degree spacing in latitude and longitude.

These codes were produced using **Python 3** (Version 3.8); The following packages have to be installed on your python: **numpy**, **scipy**, **netCDF4**, **matplotlib** and **cartopy**. You can install the packages using pip (for example "`pip install numpy`") or conda (for example "`conda install numpy`"). Note that in order to mask out points on land in the second code, the "**global-land-mask**" library is used and also has to be installed.

Note: Cartopy installation may be a bit complex as it has a lot of required dependencies. Using pre-built binaries can be a solution. They can be found at a variety of sources. Christoph Gohlke maintains unofficial Windows binaries (https://www.lfd.uci.edu/~gohlke/pythonlibs/). The correct version of the package of interest (ex: cp38 and win_amd64 is for Python 3.8 (64bits)) can be downloaded and installed by using "`pip install`*" followed by the path of the .whl file which has just been downloaded (*`pip install C:/some-dir/some-file.whl`*).*

1.2.2 Plot an unstructured grid of Baltic SEAL

```
# Importing the needed packages:

import netCDF4
import numpy as np
import cartopy as cart
from cartopy.mpl.ticker import LongitudeFormatter, LatitudeFormatter
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
from scipy.interpolate import griddata

# Setting the inputs:

directory='C:/some-dir/' #The user should change the path to the .nc files' directory
filename='2002_06.nc' #The user should change to the right file
S = netCDF4.Dataset(directory+filename)
lon = S.variables['lon'][:]
lat = S.variables['lat'][:]
ssh = S.variables['ssh'][:]
min_lat=53.0
max_lat=66.0
min_lon=9.0
max_lon=31.0


# Displaying the unstructured grid in a scatterplot:

fig = plt.plot()
plt.rcParams.update({'font.size': 15})
plt.plot
plt.rcParams["figure.figsize"] = (50,10) #Increase figure size
ax = plt.axes(projection=ccrs.Miller())
img=plt.scatter(lon, lat,
         c=ssh, s=20,
         cmap='cool', alpha=1,transform=ccrs.PlateCarree())
ax.coastlines(resolution='10m', color='black', linewidth=1)
ax.set_xticks(np.arange(min_lon,max_lon,2), crs=ccrs.PlateCarree())
ax.set_yticks(np.arange(min_lat,max_lat,1), crs=ccrs.PlateCarree())
lon_formatter = cart.mpl.ticker.LongitudeFormatter(number_format='.1f',
                                   degree_symbol='',
                                   dateline_direction_label=True)
lat_formatter = cart.mpl.ticker.LatitudeFormatter(number_format='.1f',
                                  degree_symbol='')
ax.xaxis.set_major_formatter(lon_formatter)
ax.yaxis.set_major_formatter(lat_formatter)
plt.colorbar(img,label=r'SSH (m)')
plt.clim(15, 40)
plt.show() #A window will show up, allowing the user to see and download the plot
```

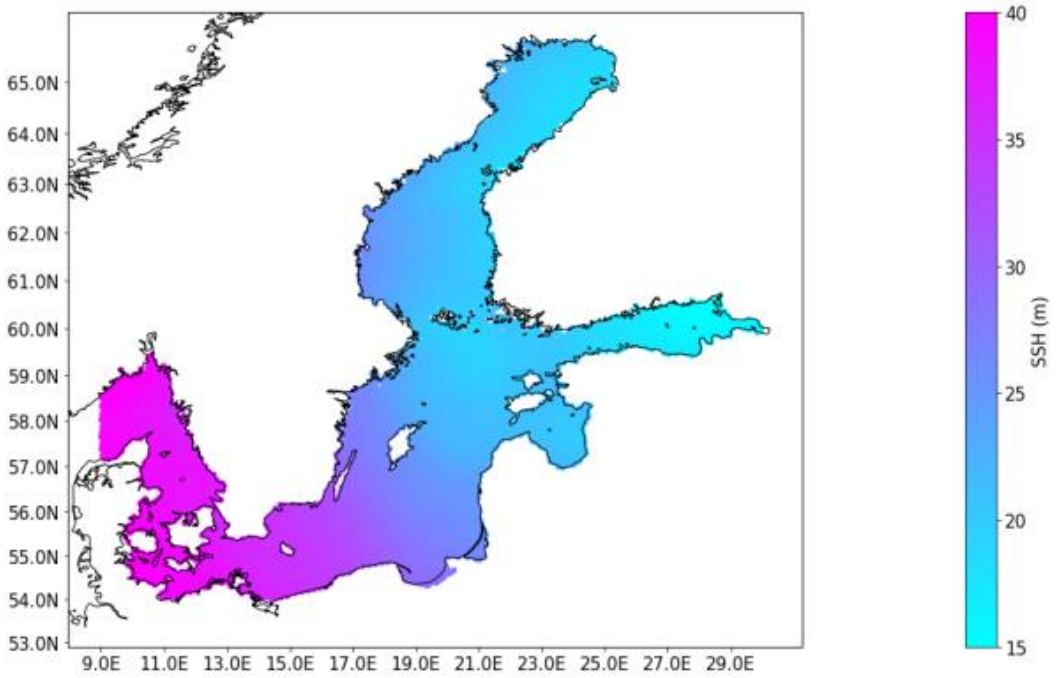

Figure 3: *Output of the first python code: Baltic SEAL unstructured grid displayed in a scatter plot*

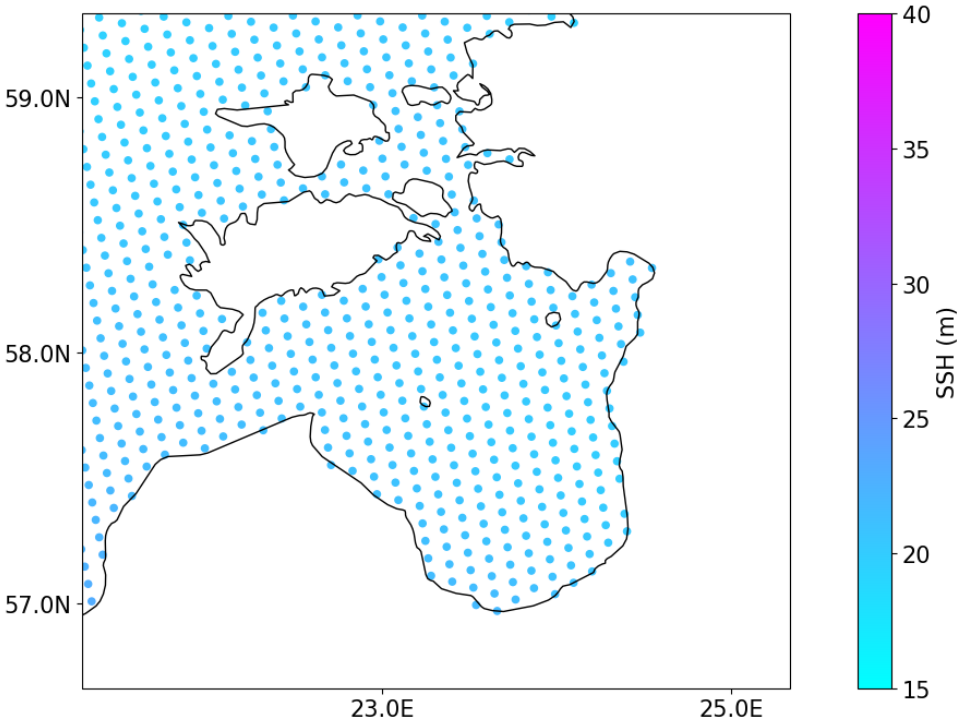

Figure 4: *Output of the first python code: Zoom on the Gulf of Riga*

1.2.3 Plot a structured grid of Baltic SEAL

```
# Importing the needed packages:
import netCDF4
import numpy as np
import cartopy as cart
from cartopy.mpl.ticker import LongitudeFormatter, LatitudeFormatter
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
from scipy.interpolate import griddata
from global_land_mask import globe

# Setting the inputs:
directory='C:/some-dir/' #The user should change the path to the .nc files' directory
filename='2002_06.nc' #The user should change to the right file
S = netCDF4.Dataset(directory+filename)
lon = S.variables['lon'][:]
lat = S.variables['lat'][:]
ssh = S.variables['ssh'][:]
min_lat=53.0
max_lat=66.0
min_lon=9.0
max_lon=31.0

grid_size=0.1 #The user can change the grid size to the desired one
grid_x, grid_y = np.mgrid[max_lat:min_lat:-grid_size,min_lon:max_lon:grid_size]
points=np.column_stack((lon,lat))
grid_z0 = griddata(points, ssh, (grid_y, grid_x), method='linear')

# Displaying an interpolated Baltic SEAL product onto a structured grid:

fig = plt.plot()
plt.rcParams.update({'font.size': 15})
plt.plot
plt.rcParams["figure.figsize"] = (50,10) #Increase figure size
ax = plt.axes(projection=ccrs.Miller())
for i in np.arange(0,np.shape(grid_y)[0]) :
   for j in np.arange(0,np.shape(grid_y)[1]) :
       if globe.is_land( grid_x[i,j],grid_y[i,j]) :
           grid_z0[i,j] = np.nan
img=plt.pcolormesh(grid_y,grid_x,grid_z0,
          cmap='cool', alpha=1,transform=ccrs.PlateCarree())
ax.coastlines(resolution='10m', color='black', linewidth=1)
ax.set_xticks(np.arange(min_lon,max_lon,2), crs=ccrs.PlateCarree())
ax.set_yticks(np.arange(min_lat,max_lat,1), crs=ccrs.PlateCarree())
lon_formatter = cart.mpl.ticker.LongitudeFormatter(number_format='.1f',
                                   degree_symbol='',
                                   dateline_direction_label=True)
lat_formatter = cart.mpl.ticker.LatitudeFormatter(number_format='.1f',
                                  degree_symbol='')
ax.xaxis.set_major_formatter(lon_formatter)
ax.yaxis.set_major_formatter(lat_formatter)
plt.colorbar(img,label=r'SSH (m)')
plt.show()#A window will show up, allowing the user to see and download the plot
```

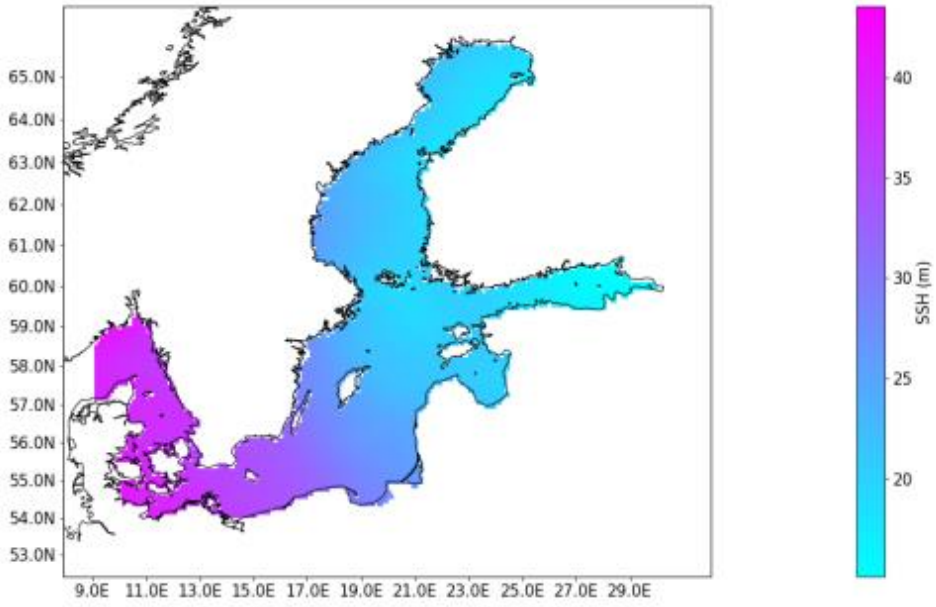

Figure 5: *Output of the second python code: interpolated Baltic SEAL product onto a structured grid*

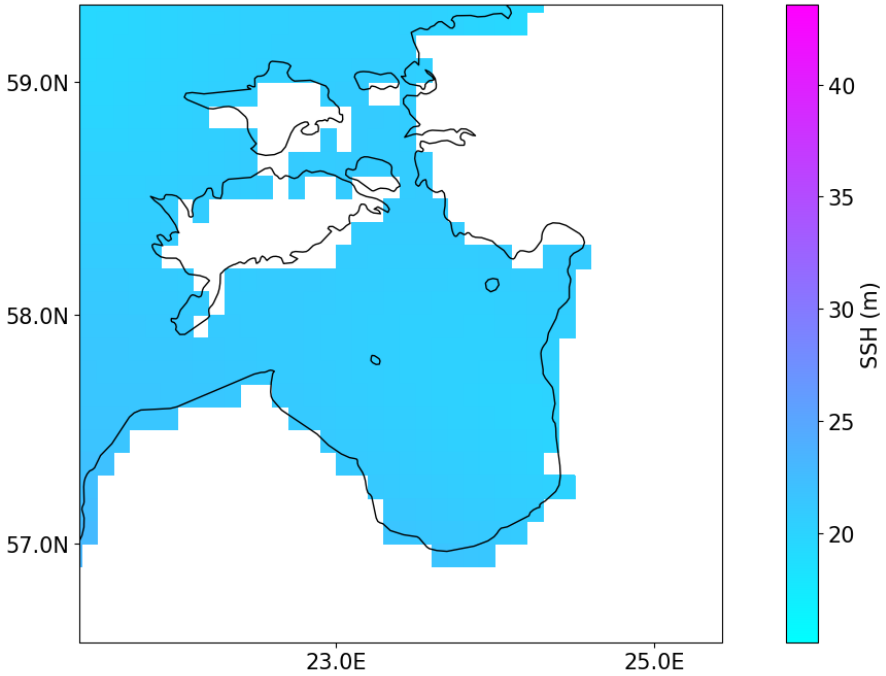

Figure 6: *Output of the second python code: Zoom on the Gulf of Riga*

3. Converting Baltic+ SEAL NetCDF4 files

3.1 How to get a Baltic+ SEAL shapefile?

This code is a sample plot for users to obtain a shapefile from along-track or unstructured grids NetCDF4 files

These codes were produced using **Python 3** (Version 3.8); The following packages have to be installed on your python: **numpy**, **os**, **netCDF4**, **gdal, csv,** and **osgeo**. You can install the packages using pip (for example "`pip install numpy`") or conda (for example "`conda install numpy`").

```
# Importing the needed packages:

import os
import netCDF4
import gdal
import numpy as np
import csv
from osgeo import osr
from osgeo import ogr

# Setting the inputs:

directory='C:/some-dir/' #the user should change the path to the .nc files' directory
filename='jason1_em_hf_262_0014.nc' #the user should change to the right file
output_directory= C:/some-dir/' #the user should change the path to the output
directory
S = netCDF4.Dataset(directory+filename)
lon = S.variables['lon'][:]
lat = S.variables['lat'][:]
ssh = S.variables['ssh'][:]

# Creation of the dictionary:

to_be_written = {}
for i in range (len(lat)):
    key = i
    if np.isnan(ssh[i]):
        to_be_written[key] =[lat[i], lon[i], 99999.999999999999999]
    else:
        to_be_written[key] =[lat[i], lon[i], ssh[i]]


print ("Dictionary written")

# Dictionary written in a csv file:

csv_file = open((output_directory+filename[0:-3])+ ".csv", "w")
headers =["ID", "ssh", "lat", "lon"]
writer = csv.DictWriter(csv_file, fieldnames= headers)
```

```
writer.writeheader()
for key, value in to_be_written.items():
    writer.writerow({'ID' : key, 'ssh': value [2], "lat": value[0], "lon":
value[1]})
csv_file.close()

print ("CSV file created")

# Create shapefile

csvfile = (output_directory + filename[0:-3]) + ".csv"
shpfile = (output_directory + filename[0:-3]) + ".shp"

spatialReference = osgeo.osr.SpatialReference()
spatialReference.ImportFromEPSG(int(4326)) #the user should change to the right EPSG
S = netCDF4.Dataset(directory+filename)
driver = osgeo.ogr.GetDriverByName('ESRI Shapefile')
shapeData = driver.CreateDataSource(shpfile)
layer = shapeData.CreateLayer('layer', spatialReference, osgeo.ogr.wkbPoint)
layer_defn = layer.GetLayerDefn()
index = 0


with open(csvfile, 'r') as csvfile:
    readerDict = csv.DictReader(csvfile, delimiter=',')
    for field in readerDict.fieldnames:
        new_field = ogr.FieldDefn(field, ogr.OFTString)
        layer.CreateField(new_field)
    for row in readerDict:
        point = osgeo.ogr.Geometry(osgeo.ogr.wkbPoint)
        point.AddPoint(float(row['lon']), float(row['lat']))
        feature = osgeo.ogr.Feature(layer_defn)
        feature.SetGeometry(point)
        feature.SetFID(index)
        for field in readerDict.fieldnames:
            i = feature.GetFieldIndex(field)
            feature.SetField(i, row[field])
        layer.CreateFeature(feature)
        index += 1
shapeData.Destroy()

print ("SHP file created")
```

3.2 How to get a Baltic+ SEAL raster?

This code is a sample plot for users to obtain a raster from unstructured grids NetCDF4 files

These codes were produced using **Python 3** (Version 3.8); The following packages have to be installed on your python: **numpy**, **netCDF4**, **gdal, scipy.interpolate, global_land_mask** and **osgeo**. You can install the packages using pip (for example "`pip install numpy`") or conda (for example "`conda install numpy`").

```
# Importing the needed packages:

import netCDF4
import numpy as np
import gdal
from scipy.interpolate import griddata
from global_land_mask import globe
from osgeo import osr
from osgeo import ogr

# Setting the inputs:

directory='C:/some-dir/'#the user should change the path to the .nc files' directory
filename='YYYY_MM.nc' #the user should change to the right file
S = netCDF4.Dataset(directory+filename)
lon = S.variables['lon'][:]
lat = S.variables['lat'][:]
ssh = S.variables['ssh'][:]
min_lat=53.0
max_lat=66.0
min_lon=9.0
max_lon=31.0

# Creating the array:

grid_size=0.1 #the user can change the size of the pixel
grid_y, grid_x = np.mgrid[max_lat:min_lat:-grid_size,min_lon:max_lon:grid_size]
points=np.column_stack((lon,lat))
grid_z0 = griddata(points, ssh, (grid_x, grid_y), method='linear')

print ("The array is created")

# Masking the land:

for i in np.arange(0,np.shape(grid_x)[0]) :
   for j in np.arange(0,np.shape(grid_x)[1]) :
       if globe.is_land( grid_y[i,j],grid_x[i,j]) :
           grid_z0[i,j] = np.nan

print("Land is masked")

# Creating the raster:
```

```
rasterName = directory + filename[0:- 3]+ ".tif"

ncols, nrows = np.shape(grid_z0)
geotransform=(min_lon,grid_size,0,min_lat,0,grid_size)
driver = gdal.GetDriverByName('GTiff')
outputRaster= driver.Create(rasterName,nrows,ncols,1,gdal.GDT_Float64)

outputRaster.SetGeoTransform(geotransform)
grid_z0 = np.flipud(grid_z0)

outband = outputRaster.GetRasterBand(1)
outband.WriteArray(grid_z0)

srs = osr.SpatialReference()
srs.ImportFromEPSG(4326)
outputRaster.SetProjection(srs.ExportToWkt())
outputRaster.FlushCache()
outputRaster = None

print("The raster file is created")
```