pyfesom Documentation

Release 1

FESOM team

Sep 20, 2017

Content:

Python library and collection of tools for basic handling of FESOM ocean model output.

Tools are python scripts with command line interfaces that are used for quick actions with FESOM model output. For example:

```
python showme.py /path/to/mesh /path/to/file.nc salt
```

will produce a map with global spatial distribution of salinity at the surface during the first time step.

Library is a python library that contains functions for working with FESOM mesh and data. For example loading FESOM mesh can be done as simple as:

```
import pyfesom as pf
meshpath  ='/path/to/mesh/'
mesh = pf.load_mesh(meshpath)
```

Examples of tools are *showme* for quick visualization of FESOM data and *scalar2geo* for interpolation to regular lon/lat grid.

CHAPTER 1

Installation

Now we support only installation from source and recomend to use conda to install dependencies. You maight succed with usual `pip install` way, but some dependencies like *cartopy* are pretty hard to install without *conda*. The shortest way to succes consist of the following simple steps:

1. Go to Miniconda website and download Miniconda installation script for your system. We recomend to use python 2.7 version.
2. Install *Miniconda*. Don't forget to add the path of *Miniconda* instllation to your *$PATH* and relaunch your terminal.
3. Execute the following lines:

```
conda config --add channels conda-forge
conda install pandas netcdf4 cartopy basemap scipy joblib seawater matplotlib␣
↪click
```

4. Go to the folder where you want to have pyfesom and execute (you have to have git installed):

   git clone https://github.com/FESOM/pyfesom.git

Now you hopefully have all dependencies in plate and a version of *pyfesom* on your system. At this point you should be able to use *Tools*.

1.1 Tools

*Pyfesom tools <:ref:'tools'>'_* are simple python scripts that are buld with use of the pyfesom library. To run the tool you should usually execute something like this:

```
python /path/to/installation/pyfesom/tools/showme.py /path/to/mesh/ /path/to/file.nc
```

That's a lot of letters. To make live easier it is recommended for Linux and Mac OS users to create an alias for every tool. For *bash* users, edit your *.bashrc* (or .bash_profile on Mac):

```
alias showme='python /path/to/installation/pyfesom/tools/showme.py'
```

For *csh* users edit your *.cshrc*:

```
alias showme python /path/to/installation/pyfesom/tools/showme.py
```

Don't forget to *source* your configuration file afterwards.

If you setup an alias as described above the call to the *showme* tool become:

```
showme /path/to/mesh/ /path/to/file.nc
```

or with some options:

```
showme -m merc -d 100 -l -6 6 21 -b -100 20 0 65 /path/to/mesh/ /path/to/file.nc
```

It also make sence to create system variables for paths to meshes.

1.2 Library

Since we do not support the standard installation of pyfesom yet, the easiest way to use the library is to add the path with library location to the system path in the beggining of the script or Jupyter notebook:

```
import sys
sys.path.append("/path/to/installation/pyfesom/")
import pyfesom as pf
```

CHAPTER 2

Tools

Collection of mostly command line style utilites for working with FESOM ocean model output.

- *showme* - shows FESOM data interpolated to regular grid, also can show bias ro climatology.
- *showo* - shows FESOM data on original grid, usually used to have a look at small regions.
- *scalar2geo* - interpolation from FESOM unstructured grid to lon/lat grid
- *f2l* - convert fesom output to more convinient structure with layers
- pcdo - execute cdo commands in parrallel

showme

2.1 showme

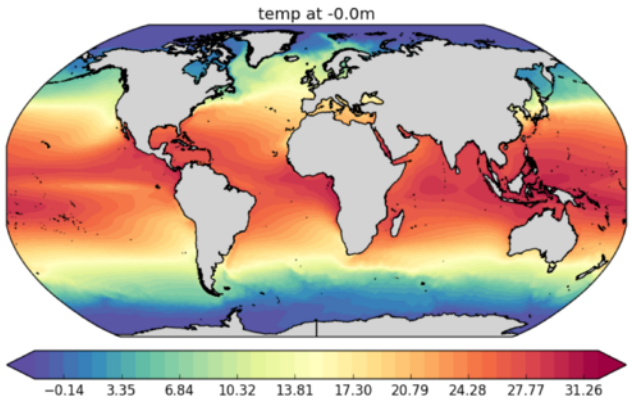
Simple visualization tool for FESOM data.

2.1.1 Basic usage

As minimum you should provide path to the mesh and path to the file:

```
python showme.py /path/to/mesh/ /path/to/file.nc
```

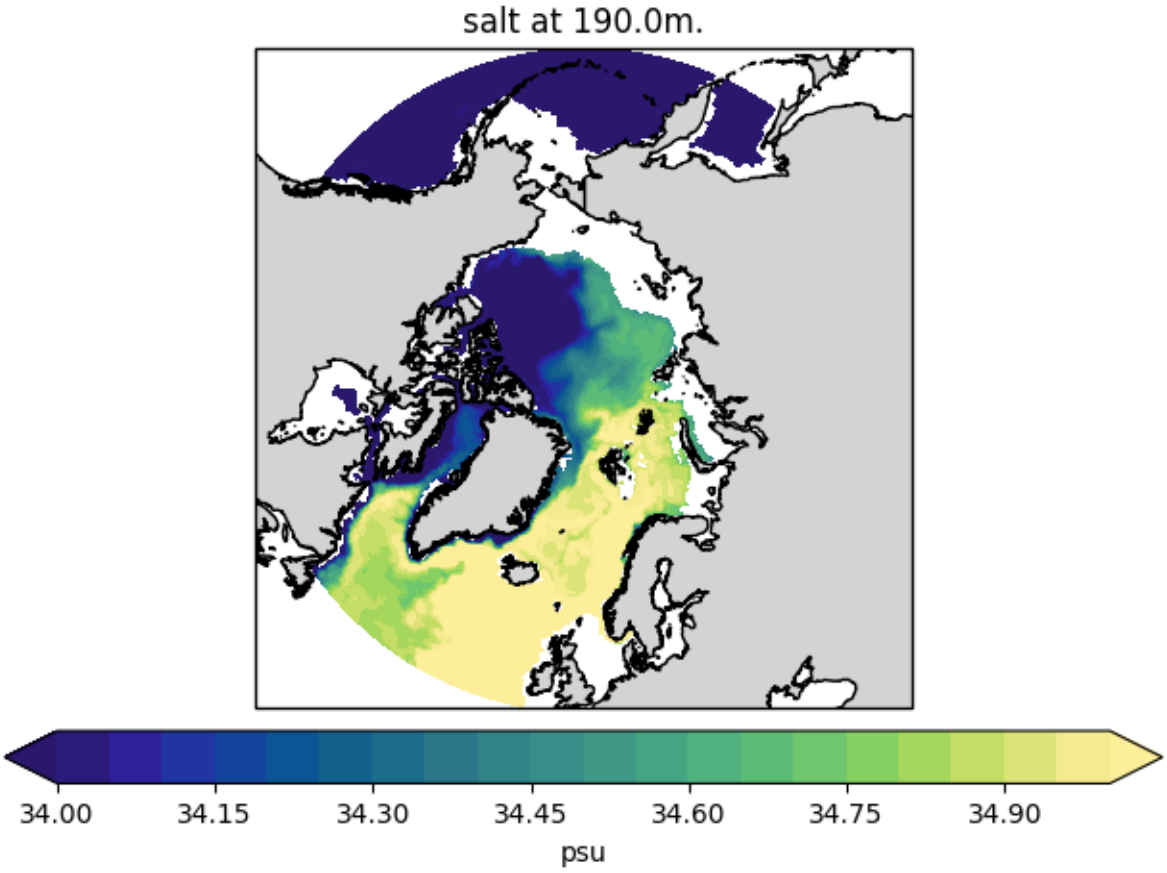
You will get an image of global distribution of temperature (*temp*) at the first time step and smallest depth:



To control different aspects of your plot you can provide additional options. For example this line:

```
python showme.py -m np -b -180 180 50 90 -d 200 -l 34 35 21 --cmap haline /path/to/
↪mesh/ /path/to/file.nc salt
```

will creaate image in the North Polar Stereo projection (*-m np*), with lon/lat boundaries of -180/180/50/90 (*-b -180 180 50 90*) ad 200 m depth (*-d 200*) with color levels spanning from 34 to 35 with 21 step (*-l 34 35 21*) and *haline* colormap from cmocean package (*–cmap haline*). The property to plot is *salt* (at the end of the line).



2.1.2 Usage and options

Below you can find complete list of options. You can allways display this list in the terminal by executing:

```
python showme.py --help
```

```
Usage: showme.py [OPTIONS] MESHPATH IFILE [VARIABLE]

meshpath - Path to the folder with FESOM1.4 mesh files.

ifile    - Path to FESOM1.4 netCDF file.

variable - The netCDF variable to be plotted.

Options:
-d, --depth FLOAT               Depth in meters.  [default: 0]
-b, --box <INTEGER RANGE INTEGER RANGE INTEGER RANGE INTEGER RANGE>...
                                Map boundaries in -180 180 -90 90 format.
                                [default: -180, 180, -80, 90]
-r, --res <INTEGER INTEGER>...  Number of points along each axis (for lon
                                and  lat).  [default: 360, 170]
-i, --influence INTEGER         Radius of influence for interpolation, in
                                meters.  [default: 80000]
-t, --timestep INTEGER          Timstep from netCDF variable, strats with 0.
                                [default: 0]
-l, --levels FLOAT...           Levels for contour plot in format min max
                                numberOfLevels. If not provided min/max
                                values from data will be used with 40
                                levels.
-q, --quiet                     If present additional information will not
                                be printed.
-o, --ofile PATH                Path to the output figure. If present the
                                image will be saved to the file instead of
                                showing it.
-m, --mapproj [merc|pc|np|sp|rob]
                                Map projection. Options are Mercator (merc),
                                Plate Carree (pc), North Polar Stereo (np),
                                South Polar Stereo (sp),  Robinson (rob)
                                [default: rob]
--abg <FLOAT FLOAT FLOAT>...    Alpha, beta and gamma Euler angles. If you
                                plots look rotated, you use wrong abg
                                values. Usually nessesary only during the
                                first use of the mesh.  [default: 50, 15,
                                -90]
-c, --clim [phc|woa05|gdem]     Select climatology to compare to. If option
                                is set the model bias to climatology will be
                                shown.
--cmap TEXT                     Name of the colormap from cmocean package or
                                from the standard matplotlib set. By default
                                `Spectral_r` will be used for property plots
                                and `balance` for bias plots.
--interp [nn|idist|linear|cubic]
                                Interpolation method. Options are nn -
                                nearest neighbor (KDTree implementation,
                                fast), idist - inverse distance (KDTree
                                implementation, decent speed), linear (scipy
                                implementation, slow) and cubic (scipy
                                implementation, slowest and give strange
```

```
                                results on corarse meshes).  [default: nn]
--ptype [cf|pcm]                Plot type. Options are contourf ('cf') and
                                pcolormesh ('pcm')  [default: cf]
-k INTEGER                      k-th nearest neighbors to use. Only used
                                when interpolation method (--interp) is
                                idist  [default: 5]
--help                          Show this message and exit.
```

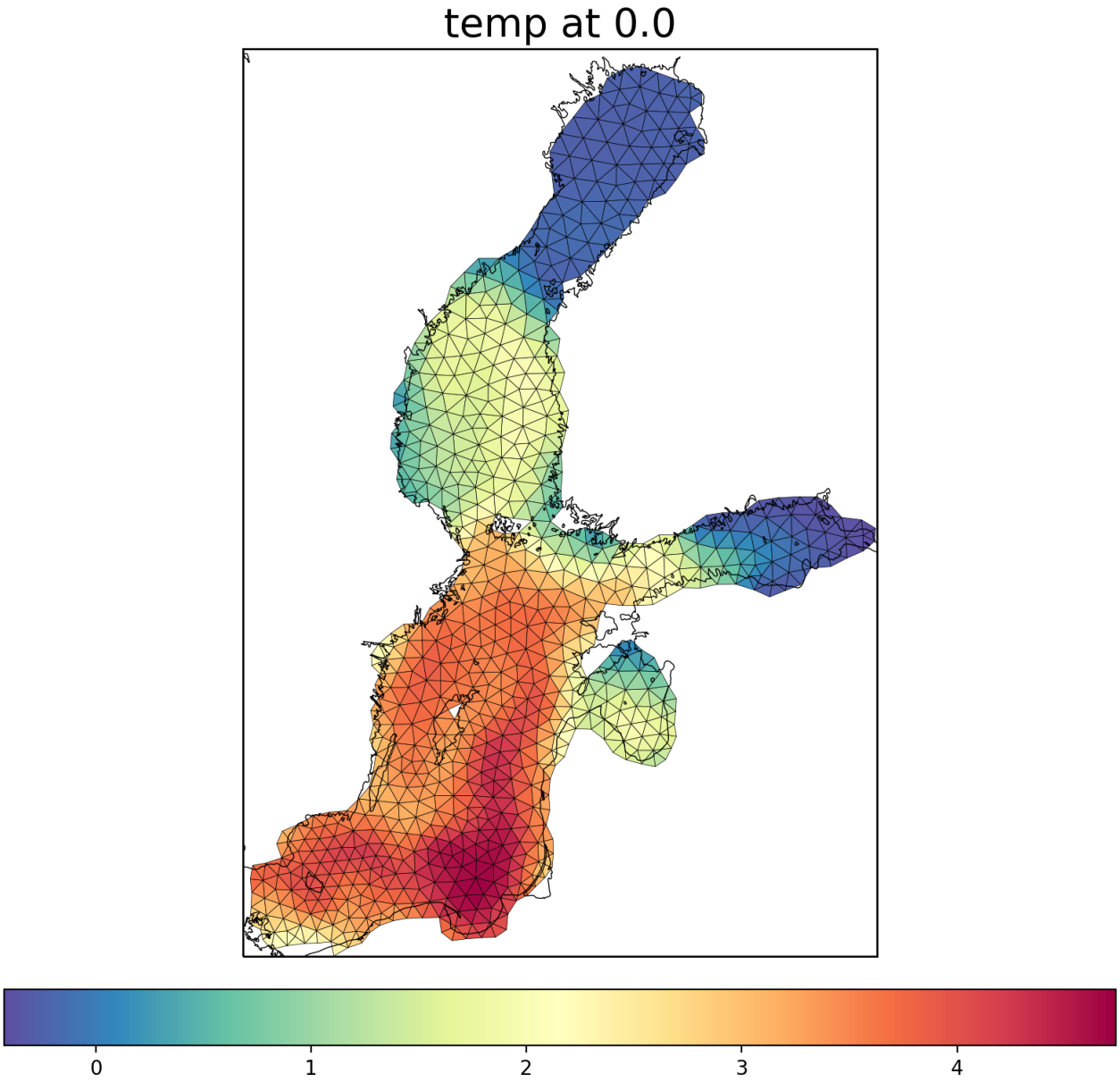
2.2 showo

Shows FESOM data on original grid.

2.2.1 Basic usage

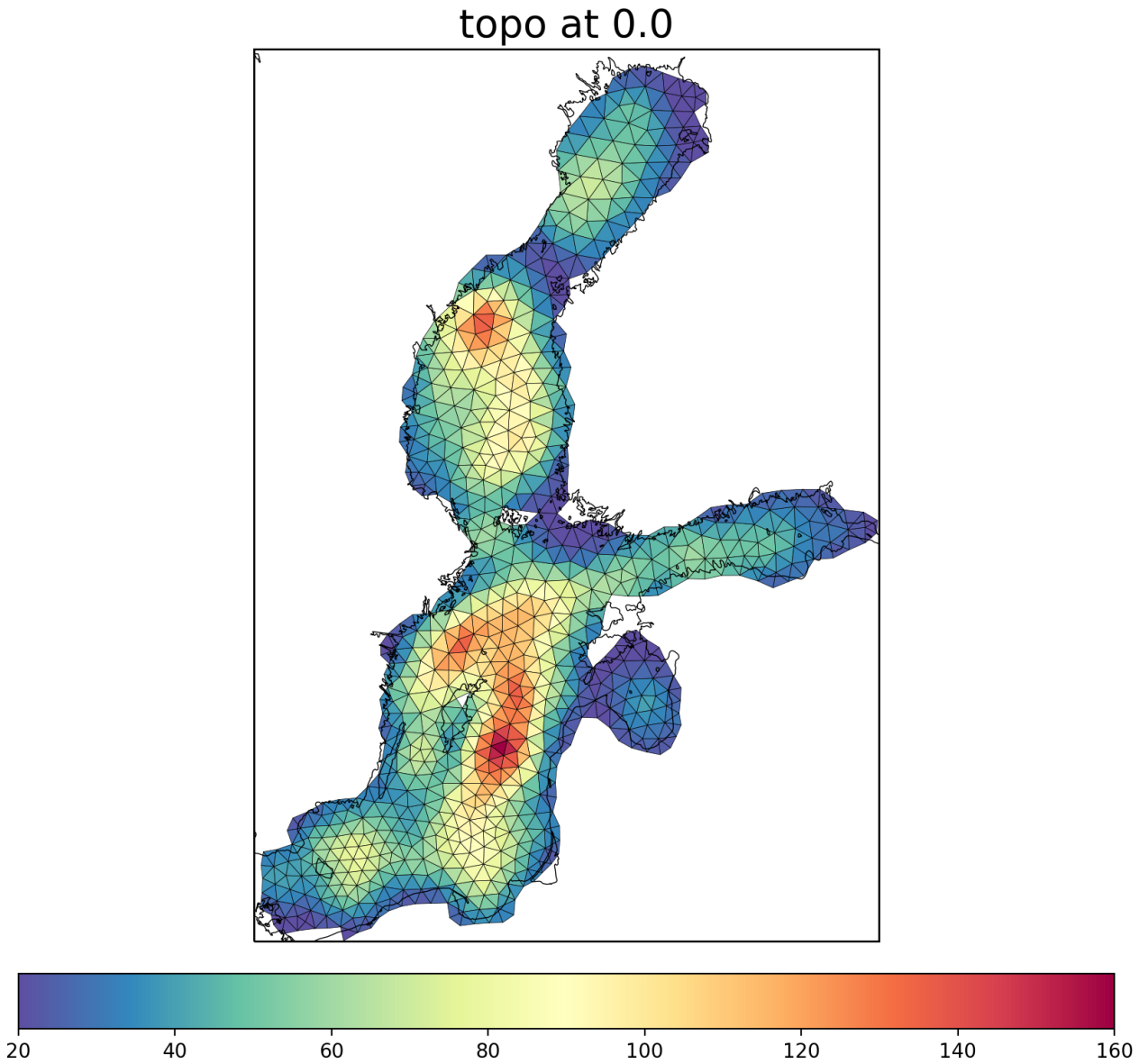
As minimum you should provide path to the mesh and path to the file:

```
python showo.py /path/to/mesh/ /path/to/file.nc
```



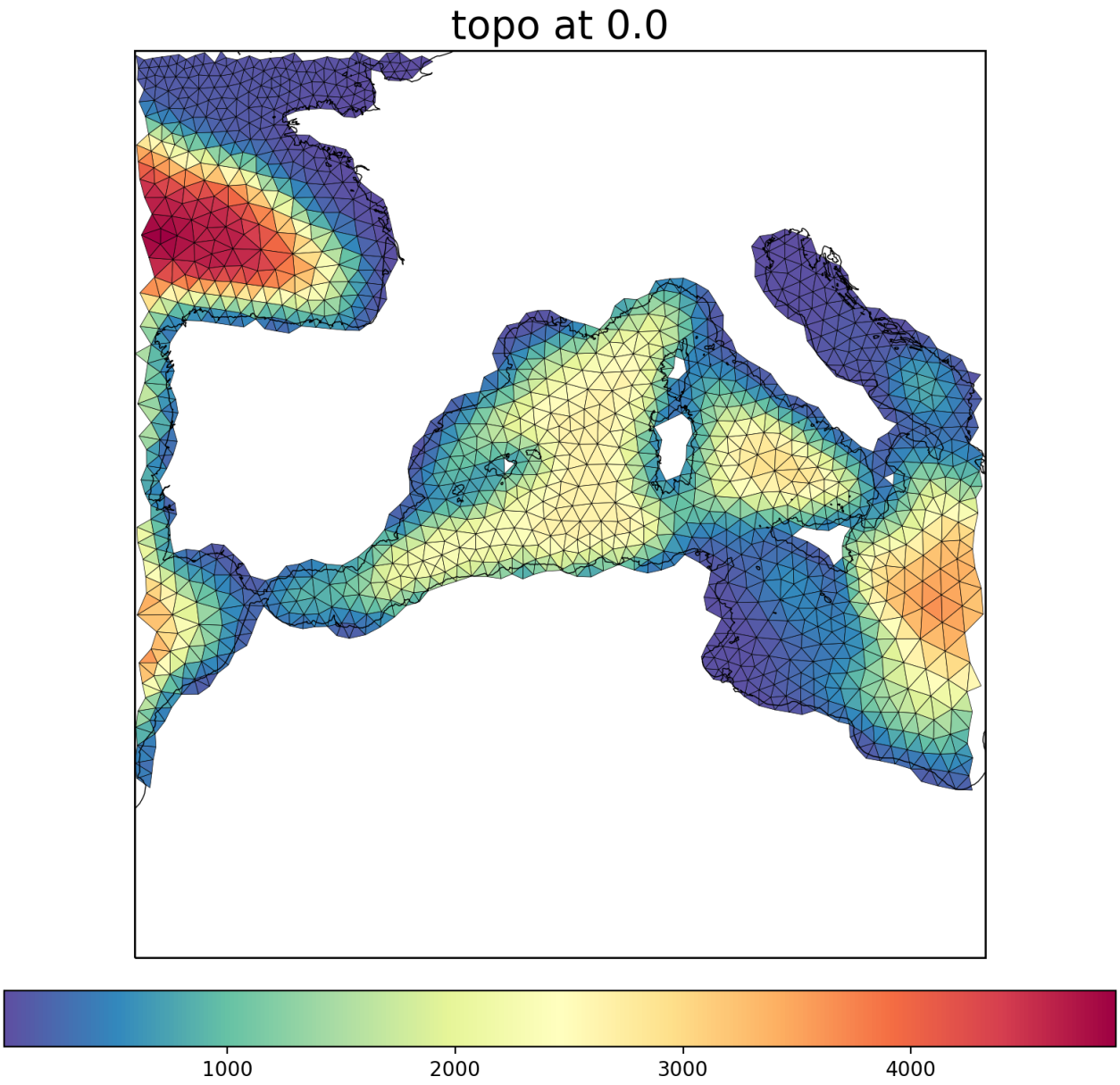
By default it will produce plot of the Baltic Sea temperature oh original grid. Quite often you youl like to plot topography on original grid - to do so, just provide *topo* as variable argument:

```
python showo.py /path/to/mesh/ /path/to/file.nc topo
```



One can easliy change the region and get the following result:

```
python showo.py -b -10 20 25 50 /path/to/mesh/ /path/to/file.nc topo
```



2.2.2 Usage and options

Below you can find complete list of options. You can allways display this list in the terminal by executing:

```
python showo.py --help
```

```
Usage: showo.py [OPTIONS] MESHPATH IFILE [VARIABLE]

meshpath - Path to the folder with FESOM1.4 mesh files.

ifile    - Path to FESOM1.4 netCDF file.

variable - The netCDF variable to be plotted.

Options:
```

```
-d, --depth FLOAT               Depth in meters.  [default: 0]
-b, --box <INTEGER RANGE INTEGER RANGE INTEGER RANGE INTEGER RANGE>...
                                Map boundaries in -180 180 -90 90 format.
                                [default: 13, 30, 54, 66]
-t, --timestep INTEGER          Timstep from netCDF variable, strats with 0.
                                [default: 0]
-l, --minmax INTEGER...         Minimun and maximum values for plotting.
-m, --mapproj [merc|pc|np|sp|rob]
                                Map projection. Options are Mercator (merc),
                                Plate Carree (pc), North Polar Stereo (np),
                                South Polar Stereo (sp),  Robinson (rob)
-q, --quiet                     If present additional information will not
                                be printed.
-o, --ofile PATH                Path to the output figure. If present the
                                image will be saved to the file instead of
                                showing it.
--help                          Show this message and exit.
```

2.3 scalar2geo

Interpolates scalar data from FESOM mesh to regular lon/lat grid.

2.3.1 Basic usage

As minimum you should provide path to the mesh, path to the file, path were the ouptut will be stored and variable name:

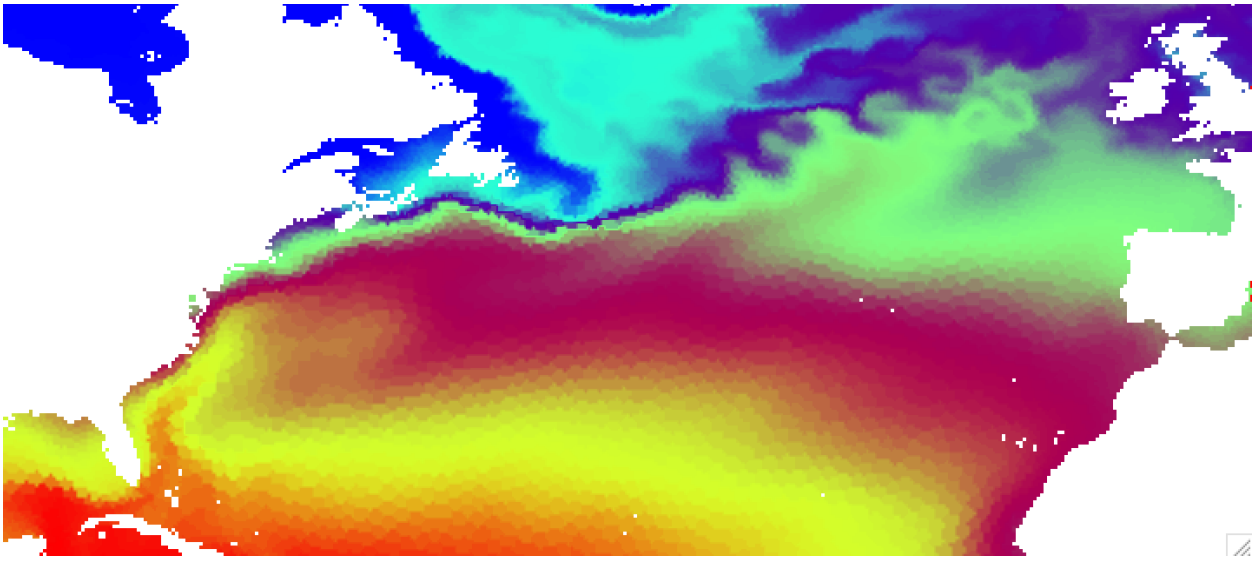
```
python scalar2geo.py  /path/to/mesh/ /path/to/file.nc /path/to/output/ temp
```

by default the field will be interpolated to the regular 1 degree grid.

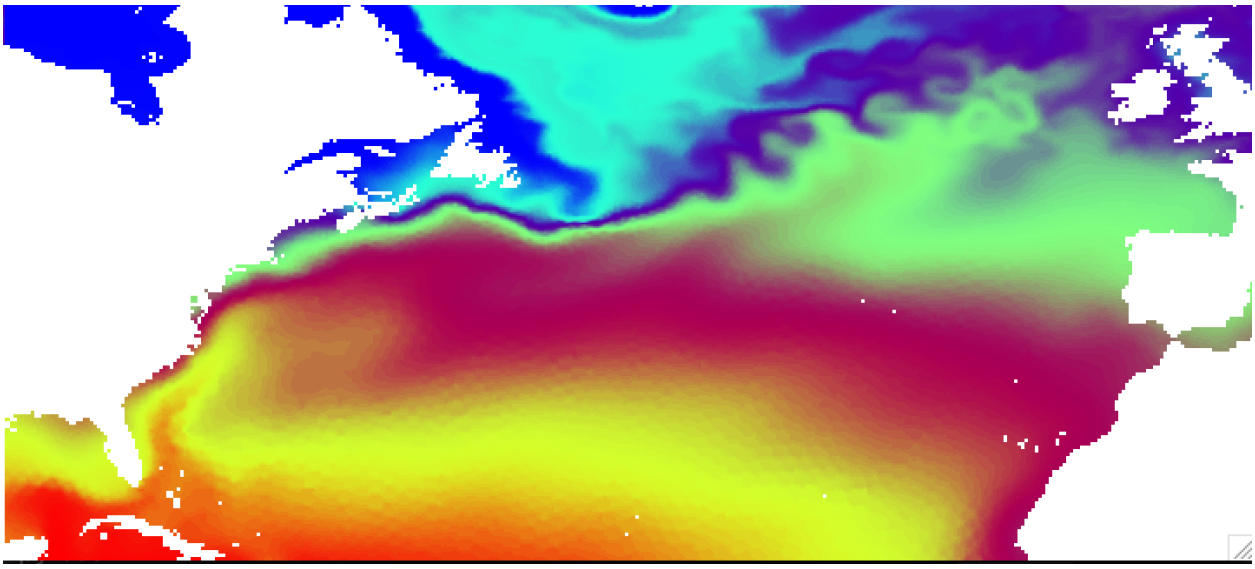
The resolution of the target grid is controlled by *-r* option, that accepts 2 arguments - number of longitudes and number of latitudes. For example to interpolate to the 1/4 degree grid for the box in the North Atlantic, you should do the following:

```
python scalar2geo.py  -b -90 0 20 60 -r 360 160 /path/to/mesh/ /path/to/file.nc /path/
↪to/output/ temp
```

If you do such interpolation for FESOM results on the COREII mesh and open resulting file in ncview, it will look like this

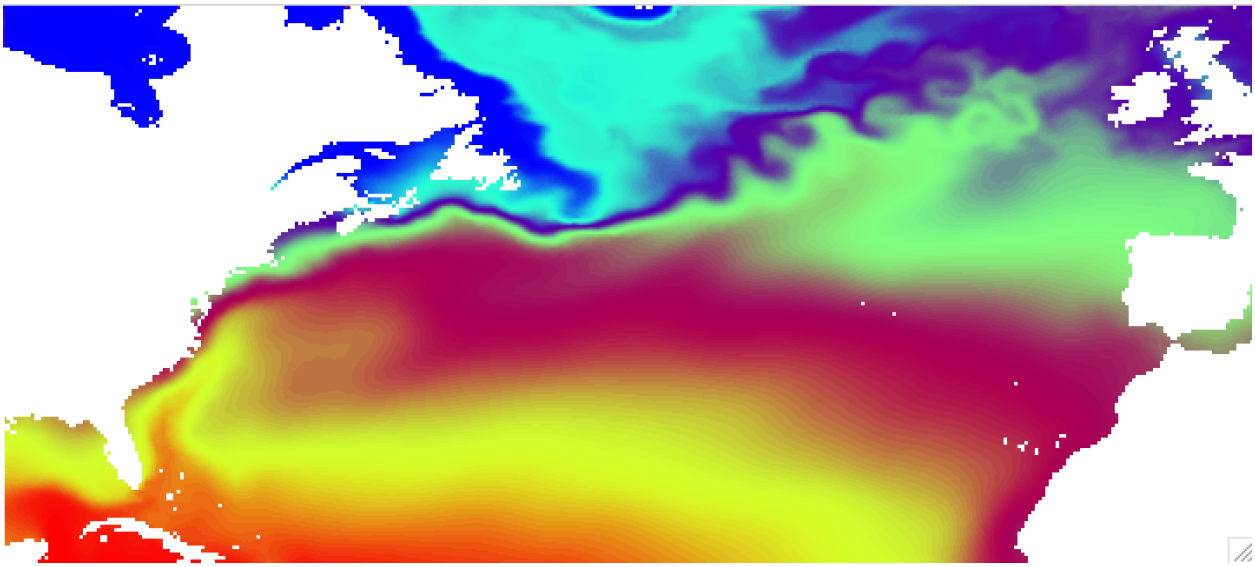


You can clearly see imprint of the original mesh on the interpolated result. This is due to the nearest neighbor interpolation used by default. The advantage of this method is that it is very fast, but for some combunations of original and target grids can produce quite ugly results. There are several other interpolation methods, namelly *idist* (inverse distance, decent speed, *linear* (scipy implementation, slow), and *cubic* (scipy implementation, slowest and give strange results on corarse meshes). The default results for *idist* method will look like this:



You can further control *idist* results with radius of influence *-i* and number of neighbors to take in to account *-k*.

The results for linear interpolation (*linear* option) will look much smoother, but interpolation will take considerably longer time:



The tool can process several files at once. You just have to specify path to your files with the wildcard like this:

```
python scalar2geo.py -n 8 /path/to/mesh/ /path/to/file_year_*.nc /path/to/output/ temp
```

By default only once processor is used, so if you would like files to be processed in parallel, you have to specify the number of parallel processes *-n*, that is usually equals to the number of processors you would like to use.

2.3.2 Usage and options

Below you can find complete list of options. You can allways display this list in the terminal by executing:

```
python showo.py --help
```

```
Usage: scalar2geo.py [OPTIONS] MESHPATH IPATH... OPATH VARIABLE

meshpath - Path to the folder with FESOM1.4 mesh files.

ipath    - Path to FESOM1.4 netCDF file or files (with wildcard).

opath    - path where the output will be stored.

variable - The netCDF variable to be converted.

Options:
-d, --depths TEXT               Depths in meters.  [default: -1]
-b, --box <INTEGER RANGE INTEGER RANGE INTEGER RANGE INTEGER RANGE>...
                                Map boundaries in -180 180 -90 90 format.
                                [default: -180, 180, -80, 90]
-r, --res <INTEGER INTEGER>...  Number of points along each axis (for lon
                                and  lat).  [default: 360, 170]
-i, --influence INTEGER         Radius of influence for interpolation, in
                                meters.  [default: 80000]
--interp [nn|idist|linear|cubic]
                                Interpolation method. Options are nn -
                                nearest neighbor (KDTree implementation,
                                fast), idist - inverse distance (KDTree
                                implementation, decent speed), linear (scipy
```

```
                                implementation, slow) and cubic (scipy
                                implementation, slowest and give strange
                                results on corarse meshes).
-t, --timestep INTEGER          Timstep from netCDF variable, starts with 0.
                                If -1, all timesteps of the netCDF file will
                                be used.  [default: 0]
--abg <FLOAT FLOAT FLOAT>...    Alpha, beta and gamma Euler angles. If you
                                plots look rotated, you use wrong abg
                                values. Usually nessesary only during the
                                first use of the mesh.
-n, --ncore INTEGER             Number of cores to use in parralel
-k INTEGER                      Number of neighbors to take in to account
                                for idist interpolation.
--help                          Show this message and exit.
```

2.4 f2l

Fesom results are saved as 1d arrays, where every value is associated with the mesh node. The connectivity (information on how to connect nodes to triangles) is usually only available for surface nodes. In order to get 2d field at other levels (to, for example, plot it) one usually create a dummy 1d array of the size equal to the number of the surface nodes and fill it with values from the model level where values exist and with NaNs where they don't (e.g. there is no level with certain depth for this surface node).

The *f2l* do exactly this - it takes 1d field that represents model 3d field and splits it in to levels, so instead of, say [time, all_3d_nodes] the dimentions become [time, level, 2d_surface_nodes]. Naturally number of nodes at the surface will be the largest among all levels, so the resulting file will be much bigger.

2.4.1 Basic usage

As minimum you should provide path to the mesh, path to the file, path were the ouptut will be stored and variable name:

```
python f2l.py /path/to/mesh/ /path/to/file.nc /path/to/output/ temp
```

The tool can process several files at once. You just have to specify path to your files with the wildcard like this:

```
python f2l.py -n 8 /path/to/mesh/ /path/to/file_year*.nc /path/to/output/ temp
```

By default only once processor is used, so if you would like files to be processed in parallel, you have to specify the number of parallel processes *-n*, that is usually equals to the number of processors you would like to use.

2.4.2 Usage and options

Below you can find complete list of options. You can allways display this list in the terminal by executing:

```
python f2l.py --help
```

```
Usage: f2l.py [OPTIONS] MESHPATH IPATH... [OPATH] [VARIABLE]

Options:
-n, --ncore INTEGER  Number of cores (parallel processes)  [default: 2]
```

```
-s, --skip            Skip the calculation if the output file already exist.
--help                Show this message and exit.
```

CHAPTER 3

Library

Pyfesom library allows you to work with FESOM meshes and data. Below you can find links to Jupyter notebooks with examples of usage. For complete description look at the *API reference* section.

3.1 Examples of library usage

- Show FESOM mesh
- Plot variable on original grid
- Plot simple diagnostics
- Interpolate to regular grid
- Compare to climatology

CHAPTER 4

API reference

4.1 Climatology

class pyfesom.climatology.**climatology**(*path*, *climname='woa05'*)

Bases: object

Class that contains information from ocean 1 degree annual climatology. Presently options are WOA2005 and PHC3.0

Parameters **path** : str

Path to the directory with climatology files

climname : str

Name of the climatology ('woa05' or 'phc')

Returns object with climatology fields

Attributes

x	(2d array) longitudes
y	(2d array) latitudes
T	(3d array) temperatures
S	(3d array) salinity
z	(1d array) depths
Tyz	(2d array) zonal mean of temperature
Syz	(3d array) zonal mean of salinity

4.2 Load mesh

pyfesom.load_mesh_data.**cut_region**(*mesh, box=[13, 30, 53, 66], depth=0*)

Cut region from the mesh.

Parameters **mesh** : object

FESOM mesh object

box : list

Coordinates of the box in [-180 180 -90 90] format. Default set to [13, 30, 53, 66], Baltic Sea.

depth : float

depth

Returns **elem_no_nan** : array

elements that belong to the region defined by *box*.

no_nan_triangles : array

boolian array of size elem2d with True for elements that belong to the region defines by *box*.

pyfesom.load_mesh_data.**fesom2depth**(*depth*, *data3*, *mesh*, *verbose=True*)

Return 2d array of the 2d mesh shape with data from the model level closest to the desired depth. There is no interpolation to the depth.

Parameters **depth** : int

desired depth

data3 : array

complete 3d data (vector) for one timestep

mesh : fesom_mesh object

mesh representation

verbose : bool

flag to turn off information about which level will be used.

Returns **data2** : array

2d array (actually vector) with data from the desired level.

class pyfesom.load_mesh_data.**fesom_mesh**(*path, abg=[50, 15, -90], get3d=True*)

Bases: object

Creates instance of the FESOM mesh.

This class creates instance that contain information about FESOM mesh. At present the class works with ASCII representation of the FESOM grid, but should be extended to be able to read also netCDF version (probably UGRID convention).

Minimum requirement is to provide the path to the directory, where following files should be located (not nessesarely all of them will be used):

- nod2d.out
- nod3d.out

- elem2d.out
- aux3d.out

Parameters **path** : str

Path to the directory with mesh files

abg : list

alpha, beta and gamma Euler angles. Default [50, 15, -90]

get3d : bool

do we load complete 3d mesh or only 2d nodes.

Returns **mesh** : object

fesom_mesh object

Attributes

path	(str) Path to the directory with mesh files
x2	(array) x position (lon) of the surface node
y2	(array) y position (lat) of the surface node
n2d	(int) number of 2d nodes
e2d	(int) number of 2d elements (triangles)
n3d	(int) number of 3d nodes
nlev	(int) number of vertical levels
zlevs	(array) array of vertical level depths
voltri	(array) array with 2d volume of triangles
alpha	(float) Euler angle alpha
beta	(float) Euler angle beta
gamma	(float) Euler angle gamma

read2d()

Reads only surface part of the mesh. Useful if your mesh is large and you want to visualize only surface.

read3d()

Reads 3d part of the mesh.

pyfesom.load_mesh_data.**get_data**(*data*, *mesh*, *depth=0*, *verbose=True*)

Show data from the model level that is closest to the desired depth.

Parameters **data** : array

complete 3d data for one timestep

mesh : fesom_mesh object

mesh representation

depth : int

desired depth

verbose : bool

flag to turn off information about which level will be used.

Returns **level_data** : array

2d array (actually vector) with data from model level that is closest to the desired depth.

elem_no_nan : array

array with triangles (defined as triplets of node indexes) with not NaN elements.

pyfesom.load_mesh_data.**get_layer_mean**(*data*, *depth*, *mesh*, *timeslice=None*)

Return mean over the model depth that is closest to specified depth.

pyfesom.load_mesh_data.**ind_for_depth**(*depth*, *mesh*)

Return indexes that belong to certain depth.

Parameters **depth** : float

desired depth. Note thre will be no interpolation, the model level that is closest to desired depth will be selected.

mesh : object

FESOM mesh object

Returns **ind_depth** : 1d array

vector with the size equal to the size of the surface nodes with index values where we have data values and missing values where we don't have data values.

ind_noempty : 1d array

vector with indexes of the *ind_depth* that have data values.

ind_empty : 1d array

vector with indexes of the *ind_depth* that do not have data values.

pyfesom.load_mesh_data.**load_mesh**(*path, abg=[50, 15, -90], get3d=True, usepickle=True, usejoblib=False*)

Loads FESOM mesh

Parameters **path** : str

Path to the directory with mesh files

abg : list

alpha, beta and gamma Euler angles. Default [50, 15, -90]

get3d : bool

do we load complete 3d mesh or only 2d nodes.

Returns **mesh** : object

fesom_mesh object

pyfesom.load_mesh_data.**read_fesom_2d**(*str_id*, *months*, *years*, *mesh*, *result_path*, *runid*, *ext*, *how='mean'*)

Note: Deprecated in pyfesom 0.1 *read_fesom_2d* will be removed in future, it is replaced by *get_data*.

pyfesom.load_mesh_data.**read_fesom_3d**(*str_id*, *months*, *years*, *mesh*, *result_path*, *runid*, *ext*, *how='mean'*)

Note: Deprecated in pyfesom 0.1 *read_fesom_3d* will be removed in future, it is replaced by *get_data*.

pyfesom.load_mesh_data.**read_fesom_mesh**(*path*, *alpha*, *beta*, *gamma*, *read_diag=True*)

Note: Deprecated in pyfesom 0.1 *read_fesom_mesh* will be removed in future, it is replaced by *load_mesh*.

4.3 Regriding

pyfesom.regriding.**clim2regular**(*climatology*, *param*, *olons*, *olats*, *levels=None*, *verbose=True*, *radius_of_influence=100000*)

Interpolation of data on the regular grid to climatology for the set of levels.

Parameters **climatology: climatology object**

FESOM climatology object

param : str

name of the parameter to interpolate. Only 'T' for temperature and 'S' for salinity are currently supported.

olons : array

1d or 2d array of longitudes to interpolate climatology on.

olats : array

1d or 2d array of longitudes to interpolate climatology on.

levels : list like

list of levels to interpolate to.

Returns **xx** : 2d array

longitudes

yy : 2d array

latitudes

out_data : 2d array

array with climatology data interpolated to desired levels

pyfesom.regriding.**create_indexes_and_distances**(*mesh*, *lons*, *lats*, *k=1*, *n_jobs=2*)

Creates KDTree object and query it for indexes of points in FESOM mesh that are close to the points of the target grid. Also return distances of the original points to target points.

Parameters **mesh** : fesom_mesh object

pyfesom mesh representation

lons/lats : array

2d arrays with target grid values.

k : int

k-th nearest neighbors to return.

n_jobs : int, optional

Number of jobs to schedule for parallel processing. If -1 is given all processors are used. Default: 1.

Returns **distances** : array of floats

The distances to the nearest neighbors.

inds : ndarray of ints

The locations of the neighbors in data.

pyfesom.regriding.**fesom2clim**(*data*, *mesh*, *climatology*, *levels=None*, *verbose=True*, *how='nn'*, *k_neighbors=10*, *radius_of_influence=100000*)

Interpolation of fesom data to grid of the climatology for set of levels.

Parameters **data** : array

1d array of FESOM 3d data for one time step

mesh : mesh object

FESOM mesh object

climatology: climatology object

FESOM climatology object

levels : list like

list of levels to interpolate. At present you can use only standard levels of WOA. If levels are not specified, all standard WOA levels will be used.

how : str

Interpolation method. Options are 'nn' (nearest neighbor) and 'idist' (inverce distance)

k : int

k-th nearest neighbors to use. Only used when how==idist

radius_of_influence : int

Cut off distance in meters.

Returns **xx** : 2d array

longitudes

yy : 2d array

latitudes

out_data : 2d array

array with data interpolated to climatology level

pyfesom.regriding.**fesom2regular**(*data*, *mesh*, *lons*, *lats*, *distances=None*, *inds=None*, *how='nn'*, *k=10*, *radius_of_influence=100000*, *n_jobs=2*)

Interpolates data from FESOM mesh to target (usually regular) mesh.

Parameters **data** : array

1d array that represents FESOM data at one level.

mesh : fesom_mesh object

pyfesom mesh representation

lons/lats : array

2d arrays with target grid values.

distances : array of floats, optional

The distances to the nearest neighbors.

inds : ndarray of ints, optional

The locations of the neighbors in data.

how : str

Interpolation method. Options are 'nn' (nearest neighbor) and 'idist' (inverce distance)

k : int

k-th nearest neighbors to use. Only used when how==idist

radius_of_influence : int

Cut off distance in meters.

n_jobs : int, optional

Number of jobs to schedule for parallel processing. If -1 is given all processors are used. Default: 1.

Returns **data_interpolated** : 2d array

array with data interpolated to the target grid.

`pyfesom.regriding.`**`lon_lat_to_cartesian`**(*lon*, *lat*, *R=6371000*)

Calculates lon, lat coordinates of a point on a sphere with radius R. Taken from http://earthpy.org/interpolation_between_grids_with_ckdtree.html

Parameters **lon** : 1d array

longitudes

lat : 1d array

latitudes

R : float

radius of the sphere

Returns **x,y,z** : 1d arrays

cartesian coordinates

`pyfesom.regriding.`**`regular2clim`**(*data*, *ilons*, *ilats*, *izlevs*, *climatology*, *levels=None*, *verbose=True*)

Interpolation of data on the regular grid to climatology for the set of levels.

Parameters **data** : array

3d array of data on regular grid for one time step

ilons : array

1d or 2d array of longitudes

ilats : array

1d or 2d array of latitudes

izlevs : array

depths of the source data

climatology: climatology object

FESOM climatology object

levels : list like

list of levels to interpolate. At present you can use only standard levels of WOA. If levels are not specified, all standard WOA levels will be used.

Returns **xx** : 2d array

longitudes

yy : 2d array

latitudes

out_data : 2d array

array with data interpolated to climatology level

pyfesom.regriding.**regular2regular**(*data*, *ilons*, *ilats*, *olons*, *olats*, *distances=None*, *inds=None*, *how='nn'*, *k=10*, *radius_of_influence=100000*, *n_jobs=2*)

Interpolates from regular to regular grid. It's a wraper around *fesom2regular* that creates an object that mimic fesom mesh class and contain only coordinates and flatten the data.

Parameters **data** : array

1d or 2d array that represents gridded data at one level.

ilons : arrea

mesh : fesom_mesh object

pyfesom mesh representation

lons/lats : array

2d arrays with target grid values.

distances : array of floats, optional

The distances to the nearest neighbors.

inds : ndarray of ints, optional

The locations of the neighbors in data.

how : str

Interpolation method. Options are 'nn' (nearest neighbor) and 'idist' (inverce distance)

k : int

k-th nearest neighbors to use. Only used when how==idist

radius_of_influence : int

Cut off distance in meters.

n_jobs : int, optional

Number of jobs to schedule for parallel processing. If -1 is given all processors are used. Default: 1.

Returns **data_interpolated** : 2d array

array with data interpolated to the target grid.

4.4 Utilites

pyfesom.ut.**scalar_g2r**(*al*, *be*, *ga*, *lon*, *lat*)
Converts geographical coordinates to rotated coordinates.

Parameters **al** : float

alpha Euler angle

be : float

beta Euler angle

ga : float

gamma Euler angle

lon : array

1d array of longitudes in geographical coordinates

lat : array

1d array of latitudes in geographical coordinates

Returns **rlon** : array

1d array of longitudes in rotated coordinates

rlat : array

1d araay of latitudes in rotated coordinates

pyfesom.ut.**scalar_r2g**(*al*, *be*, *ga*, *rlon*, *rlat*)
Converts rotated coordinates to geographical coordinates.

Parameters **al** : float

alpha Euler angle

be : float

beta Euler angle

ga : float

gamma Euler angle

rlon : array

1d array of longitudes in rotated coordinates

rlat : array

1d araay of latitudes in rotated coordinates

Returns **lon** : array

1d array of longitudes in geographical coordinates

lat : array

1d array of latitudes in geographical coordinates

pyfesom.ut.**vec_rotate_r2g**(*al*, *be*, *ga*, *lon*, *lat*, *urot*, *vrot*, *flag*)
Rotate vectors from rotated coordinates to geographical coordinates.

Parameters **al** : float

alpha Euler angle

be : float

beta Euler angle

ga : float

gamma Euler angle

lon : array

1d array of longitudes in rotated or geographical coordinates (see flag parameter)

lat : array

1d array of latitudes in rotated or geographical coordinates (see flag parameter)

urot : array

1d array of u component of the vector in rotated coordinates

vrot : array

1d array of v component of the vector in rotated coordinates

flag : 1 or 0

flag=1 - lon,lat are in geographical coordinate flag=0 - lon,lat are in rotated coordinate

Returns **u** : array

1d array of u component of the vector in geographical coordinates

v : array

1d array of v component of the vector in geographical coordinates

4.5 Module contents

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index