Deltares

dfm_tools

A Python package for D-FlowFM input and output

Jelmer Veenstra

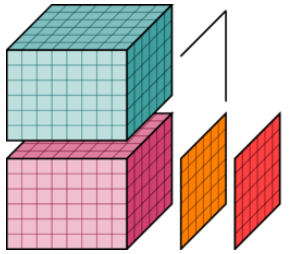
9-10-2023

dfm_tools: the concept

- What:
  - a Python package for pre- and postprocessing D-FlowFM input and output files
  - contains convenience functions built on top of other packages like xarray, xugrid, HYDROLIB-core, MeshkernelPy and many more
  - sharing of workflows among users (colleagues/external)
  - no need to re-invent the wheel

- Where:
  - code on github: https://github.com/Deltares/dfm_tools
  - installable via pip
  - jupyter notebooks and binder with example code

xarray: io of netcdf files



xarray

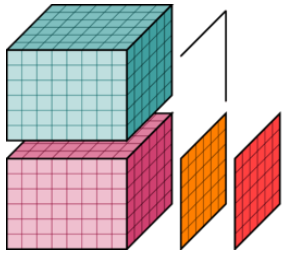
- lazy loading of netcdf files with knowledge of the entire structure
- easy indexing over all dimensions with .isel() and .sel()
- preprocess_his() enables indexing via station names (and other labels)

```
import xarray as xr
import matplotlib.pyplot as plt
plt.close('all')

from dfm_tools.xarray_helpers import preprocess_hisnc

file_nc =  r'c:\DATA\dfm_tools_testdata\DFM_3D_z_Grevelingen\\computations\\run01\\DFM_OUTPUT_Grevelingen-FM\\Grevelingen-FM_000
data_xr = xr.open_mfdataset(file_nc, preprocess=preprocess_hisnc)
```

```
Out[41]:
<xarray.Dataset>
Dimensions:                                          (stations: 24, time: 145,
                                                      laydim: 36, laydimw: 37,
                                                      source_sink_pts: 2,
                                                      source_sink: 1)
Coordinates:
    station_x_coordinate                             (stations) float64 4.747e+0...
    station_y_coordinate                             (stations) float64 4.188e+0...
    zcoordinate_c                                    (time, stations, laydim) float64 dask.array<chunksize=(145, 24, 36), meta=np.n
    zcoordinate_w                                    (time, stations, laydimw) float64 dask.array<chunksize=(145, 24, 37), meta=np.
  * time                                             (time) datetime64[ns] 2007-...
  * stations                                         (stations) <U25 'GTSO-01' ....
  * source_sink                                      (source_sink) <U2 ''
Dimensions without coordinates: laydim, laydimw, source_sink_pts
Data variables: (12/42)
    station_id                                       (stations) |S64 dask.array<chunksize=(24,), meta=np.ndarray>
    waterlevel                                       (time, stations) float64 dask.array<chunksize=(145, 24), meta=np.ndarray>
    bedlevel                                         (stations) float64 dask.array<chunksize=(24,), meta=np.ndarray>
    waterdepth                                       (time, stations) float64 dask.array<chunksize=(145, 24), meta=np.ndarray>
    x_velocity                                       (time, stations, laydim) float64 dask.array<chunksize=(145, 24, 36), meta=np.n
    y_velocity                                       (time, stations, laydim) float64 dask.array<chunksize=(145, 24, 36), meta=np.n
    ...                                               ...
```

Deltares

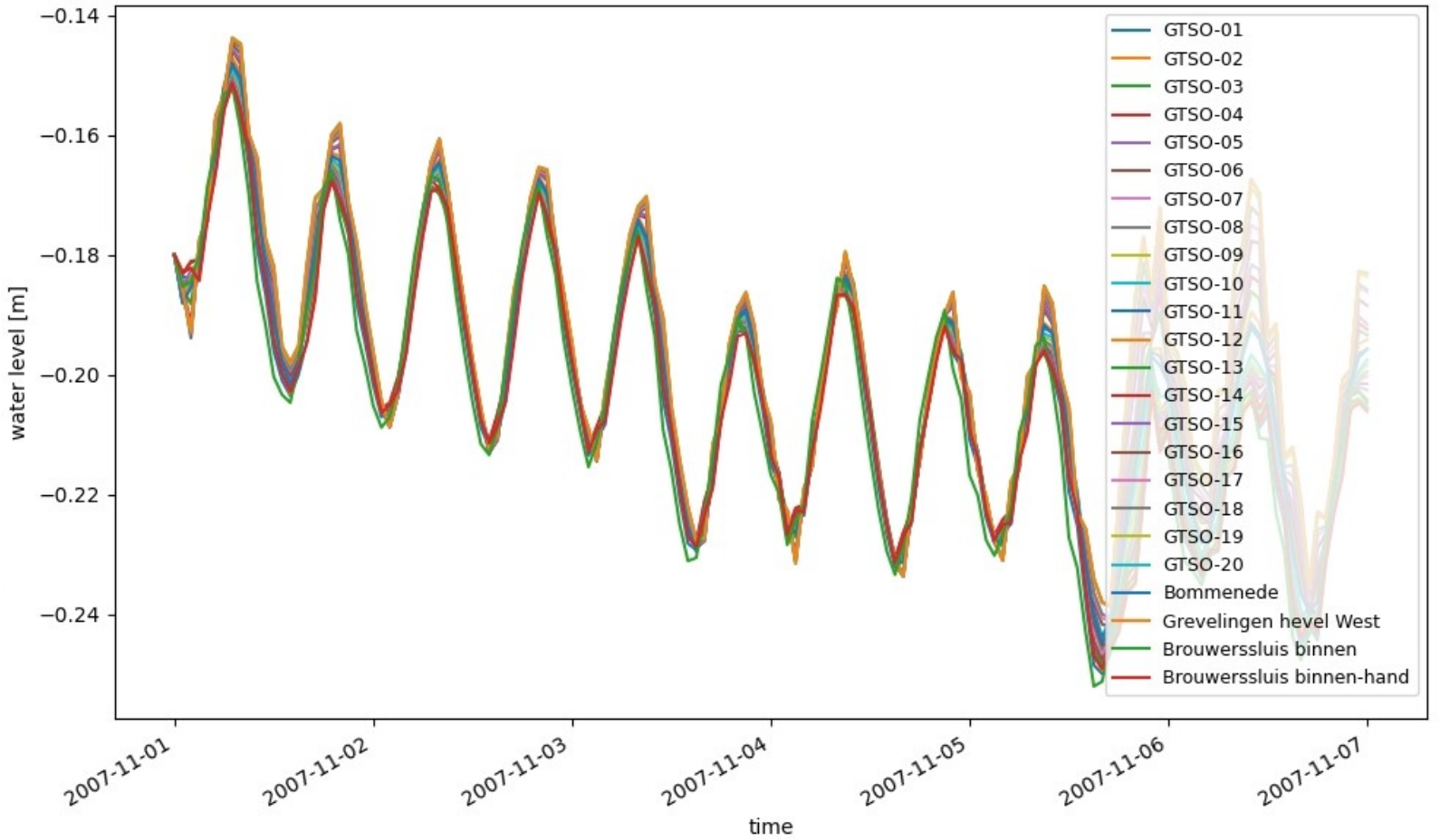
xarray: plotting of netcdf files



xarray

- easy indexing (subsetting)
- easy and complete plotting

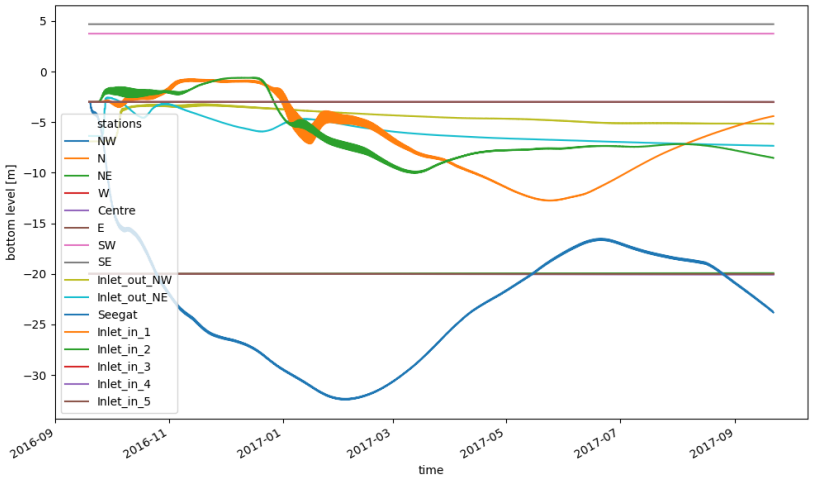
```
data_fromhis_xr = data_xr.waterlevel.sel(stations=stations_requested,time=slice('2007-11-01','2011-11-07'))
fig, ax = plt.subplots(figsize=(10,6))
data_fromhis_xr.plot.line('-',ax=ax,x='time')
```

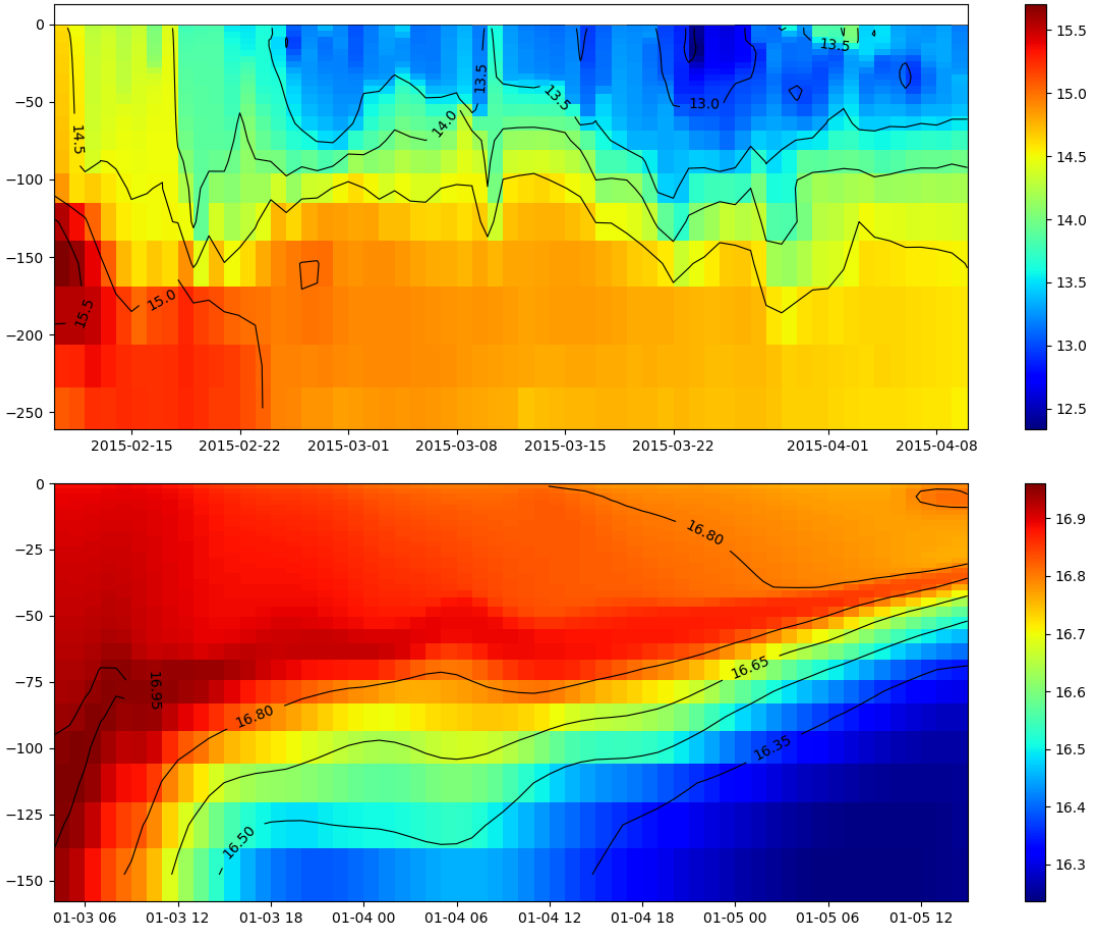


Deltares

Examples: read+plot hisfile

- xarray for reading D-FlowFM hisfile
- easy subsetting on stations, cross_sections, general structures, etc thanks to dfmt.preprocess_hisnc()
- selection of data over dimensions: time, depth, stations, cross_sections, structures, etc
- zt-plot (timeseries of one point incl. depth)

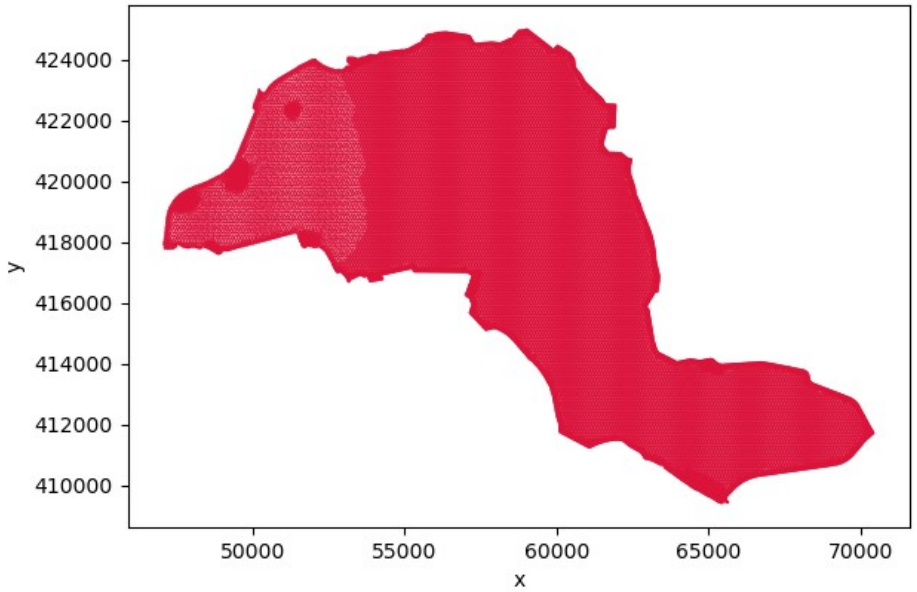


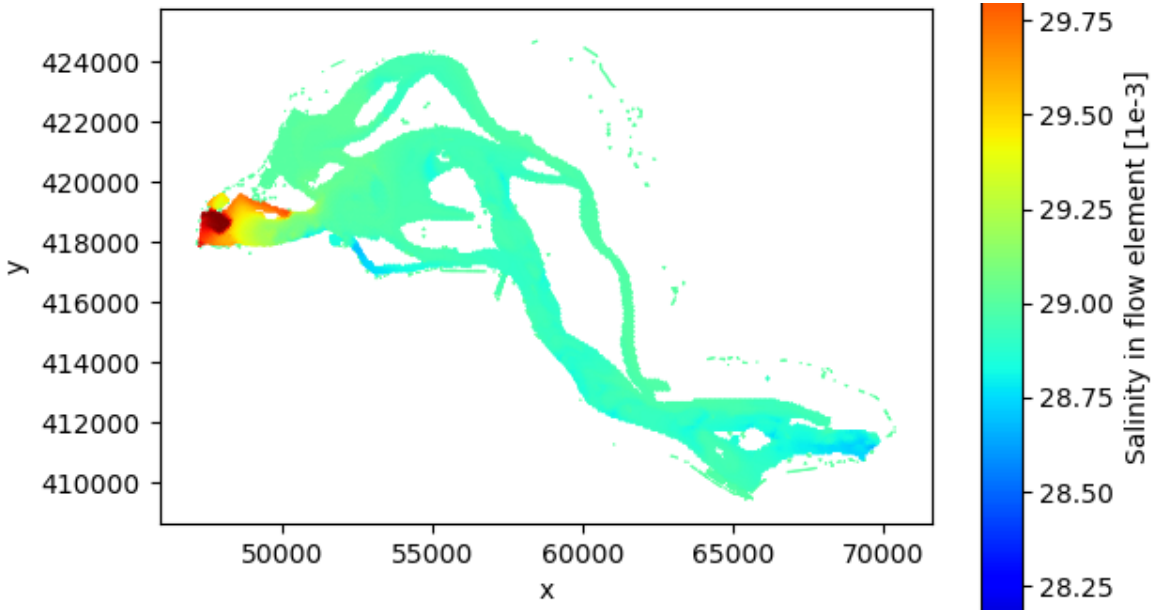


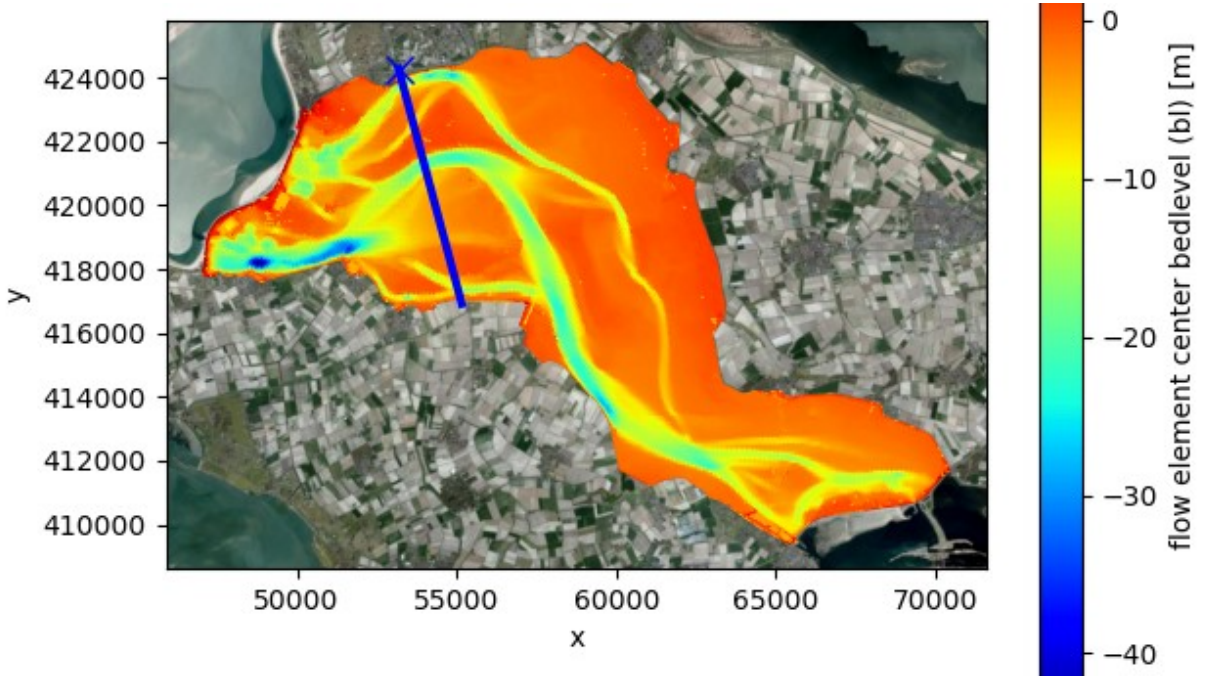
Deltares

Examples: read+plot mapfiles

- xarray/xugrid for reading D-FlowFM (unstructured) map/fou/rst/net files
- supports partitions (handling of ghost cells)
- select data based on variable and dimensions (time/layer/etc)
- satellite backgrounds with contextily

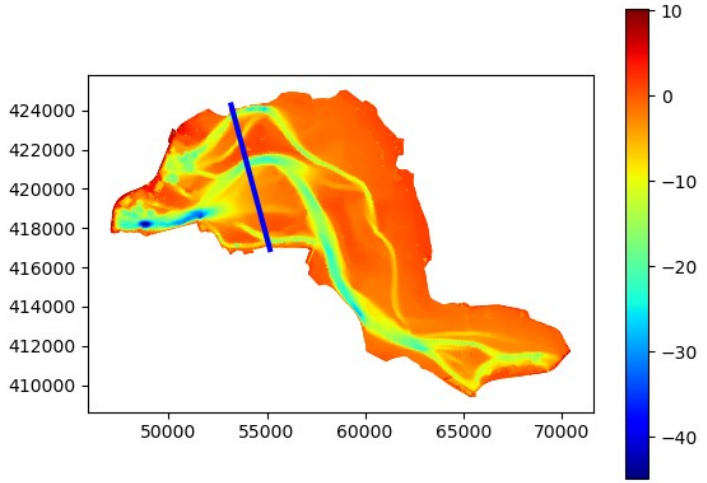


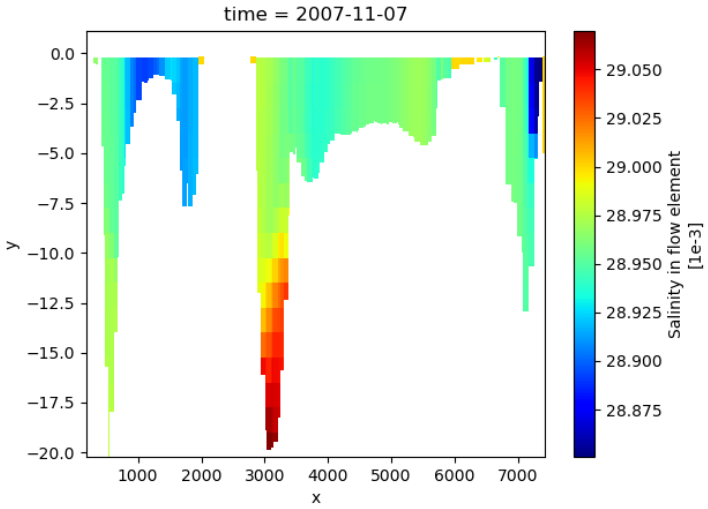


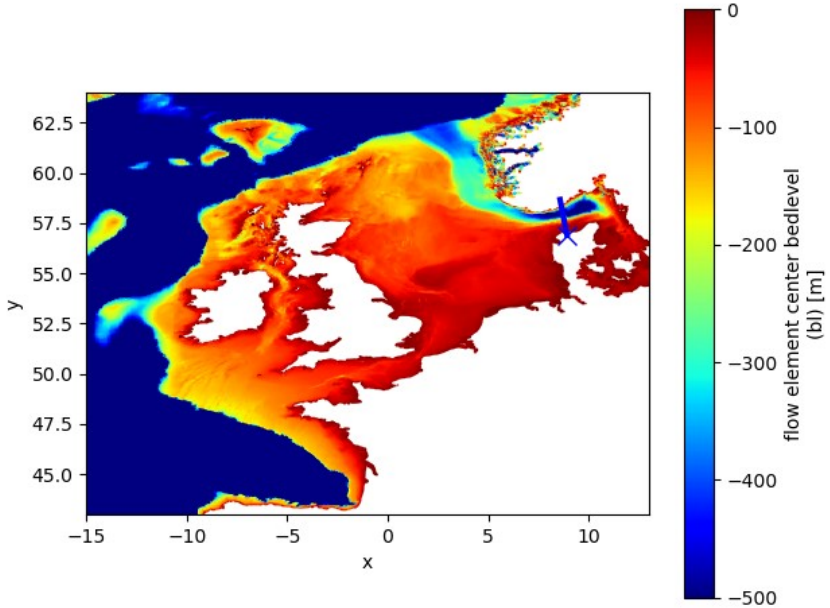


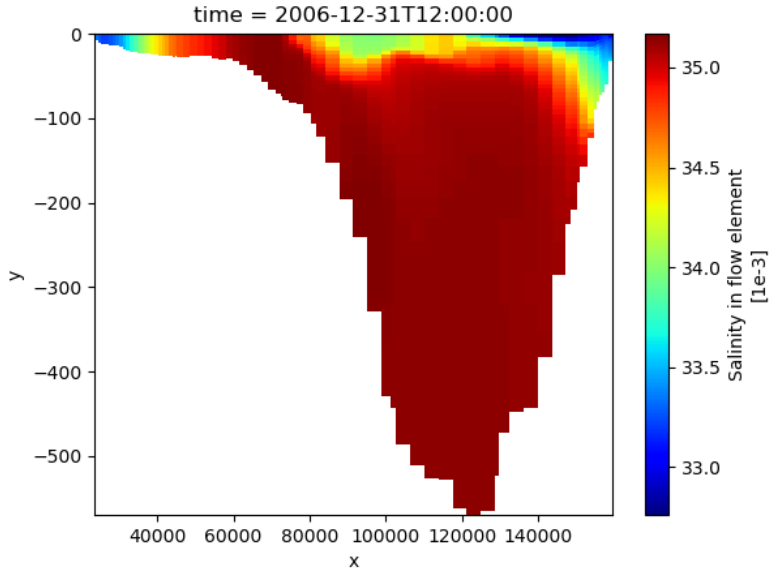
Examples: cross section trough 3D partitioned FM map data

- given a polyline (blue)
- slice trough 3D partitioned FM map data





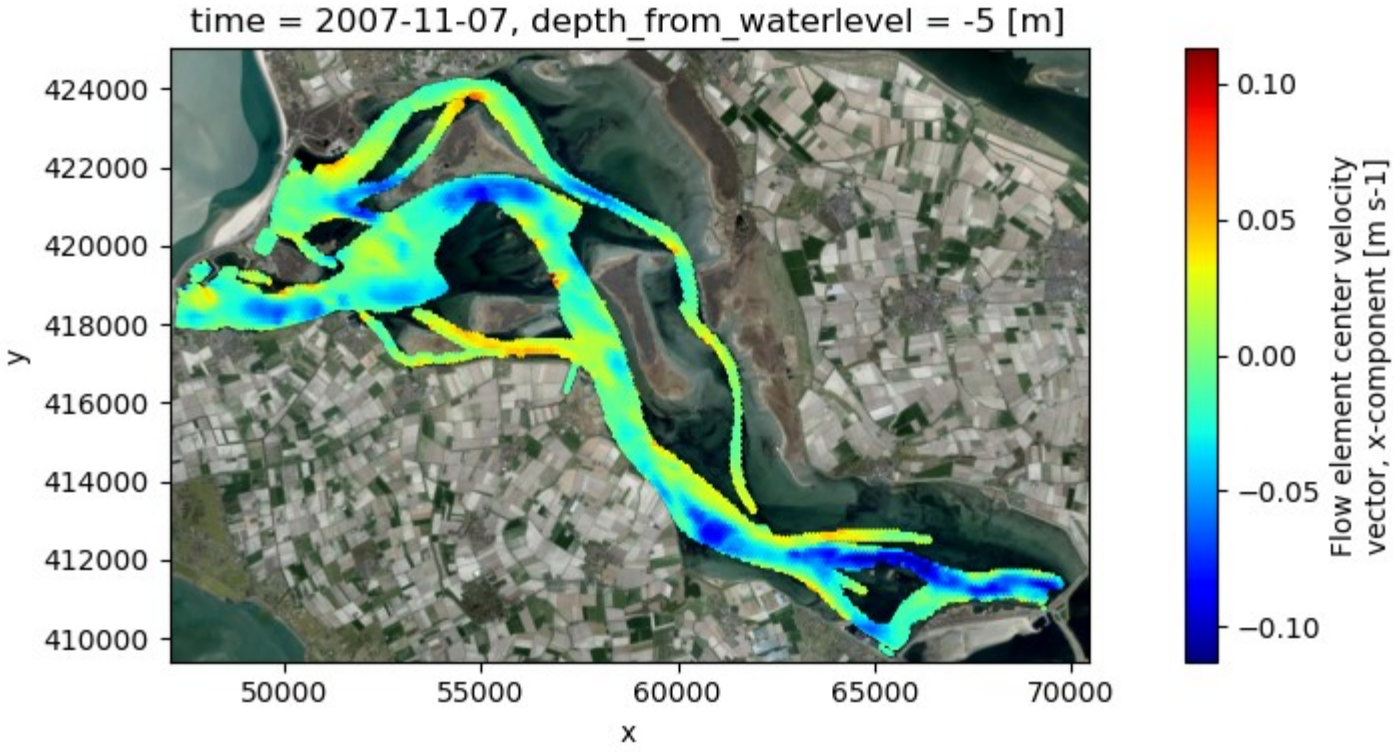




Deltares

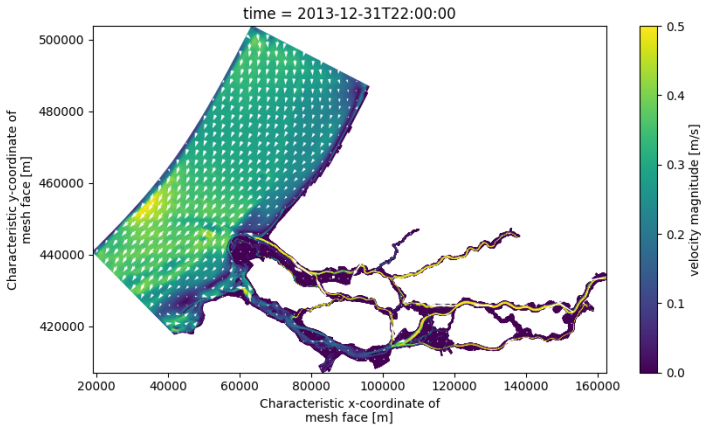
Examples: slice trough 3D partitioned FM map data

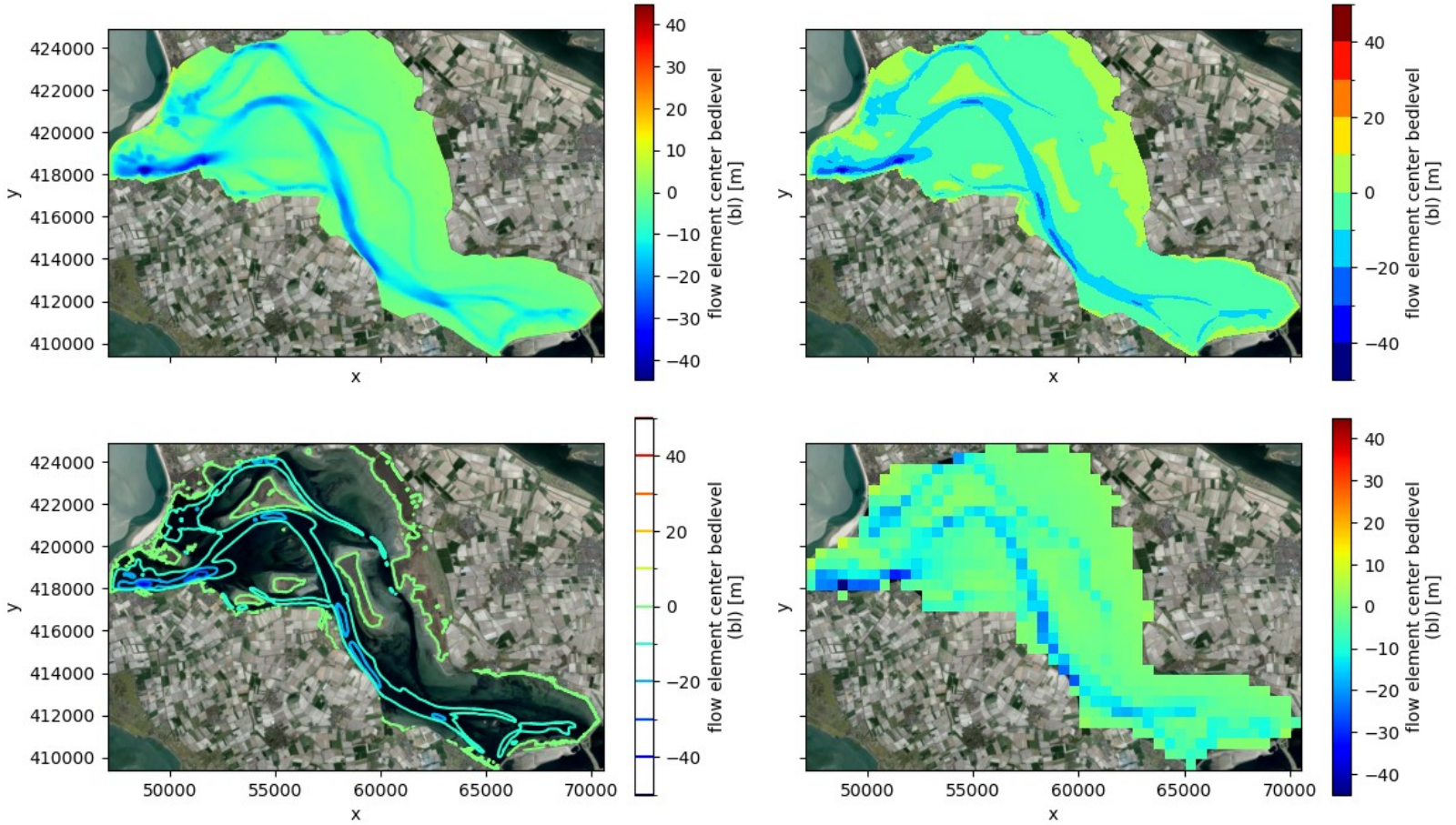
- given a depth w.r.t. z0, waterlevel, bedlevel
- slice trough 3D partitioned FM map data
- support for sigma, z and zsigma layers



Examples: various plotting options

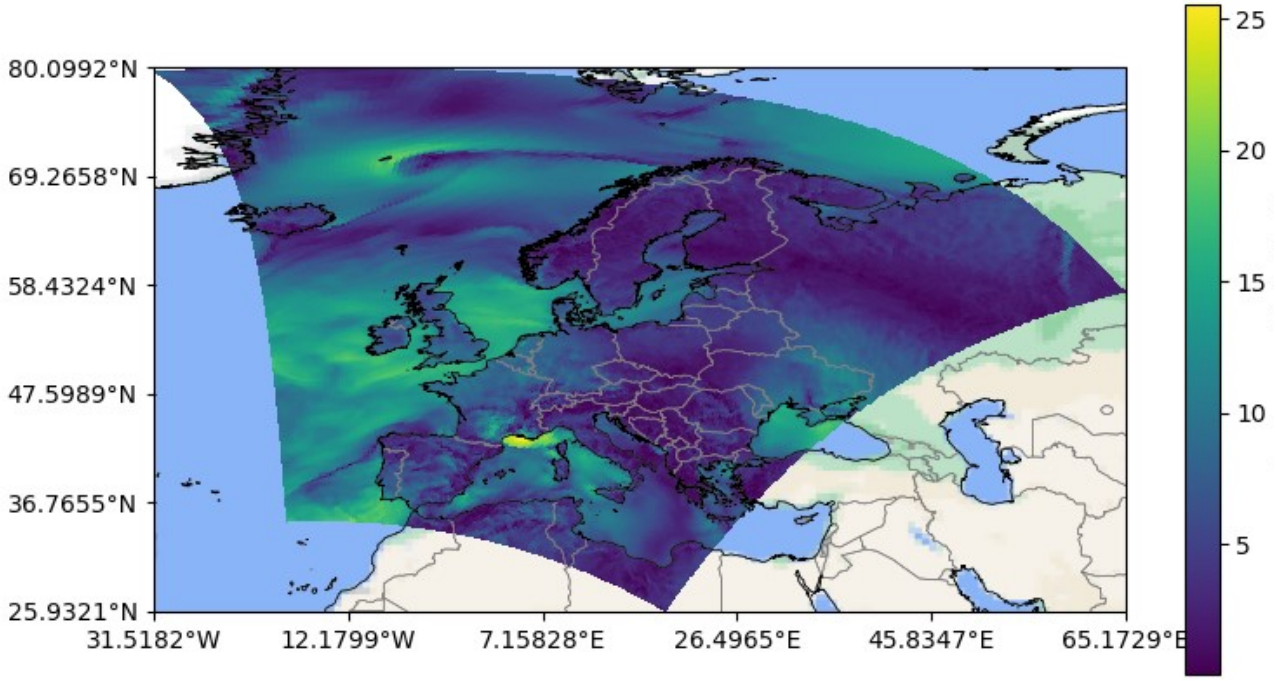
- Easy to create fancier plots like contourf or contour
- Rasterization to regular gridded dataset for e.g. different-grid comparisons
- More examples in the [xugrid docs]
- Other xugrid features: selection, interpolation, regridding, reindex_like and many more

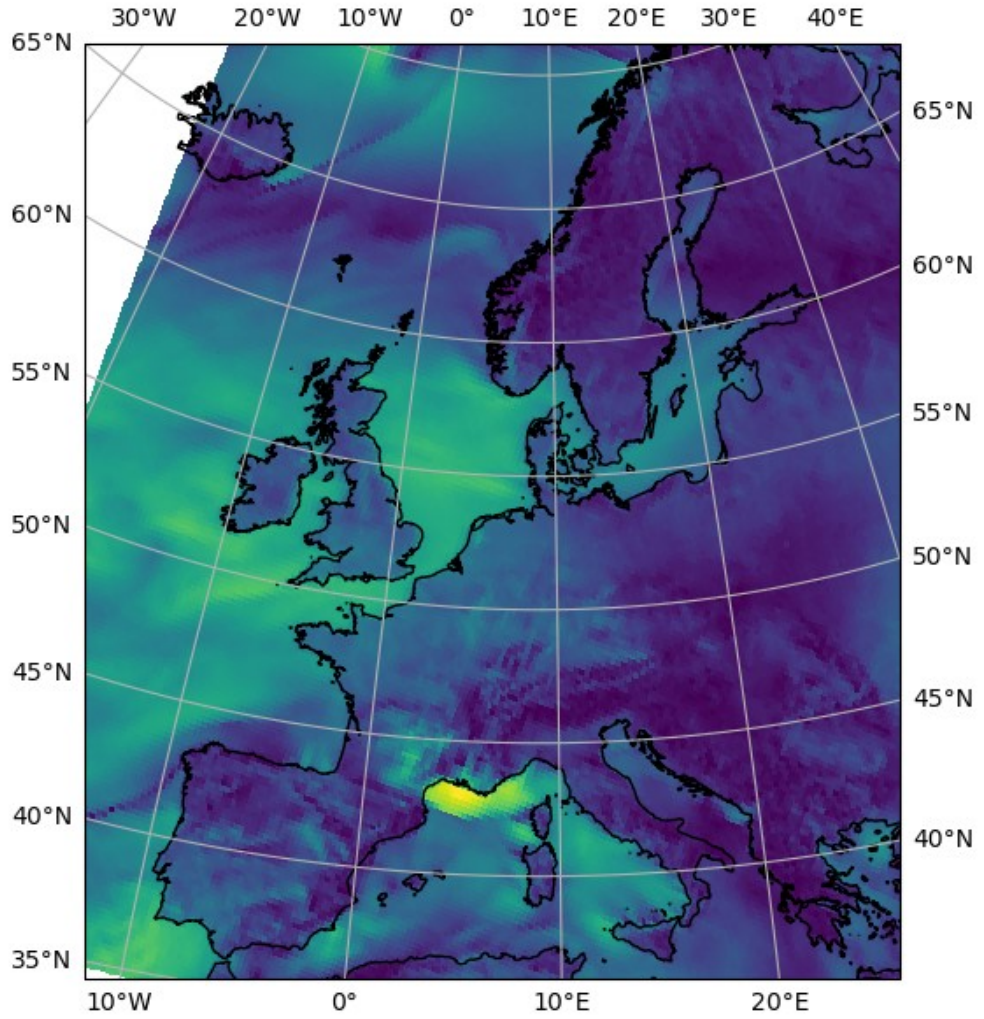




Examples: cartopy basemaps and coastlines/borders

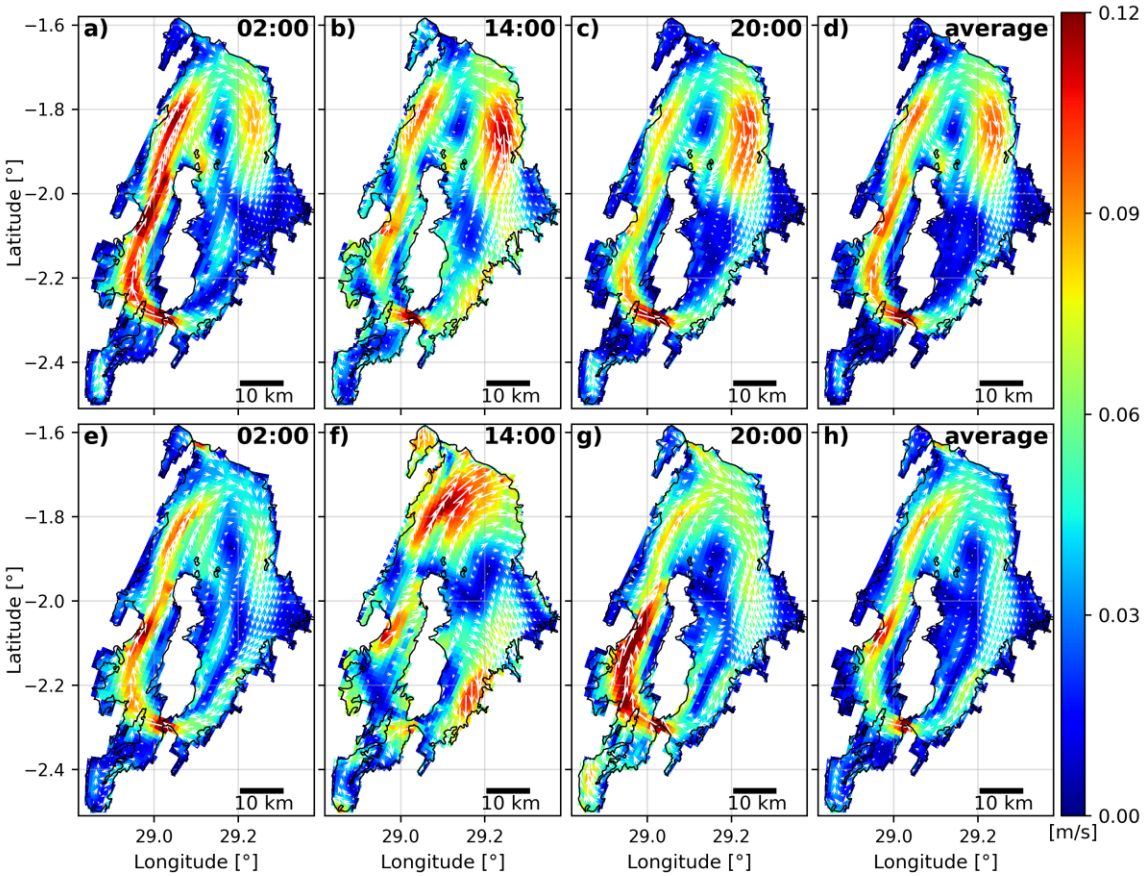
- support for almost any epsg

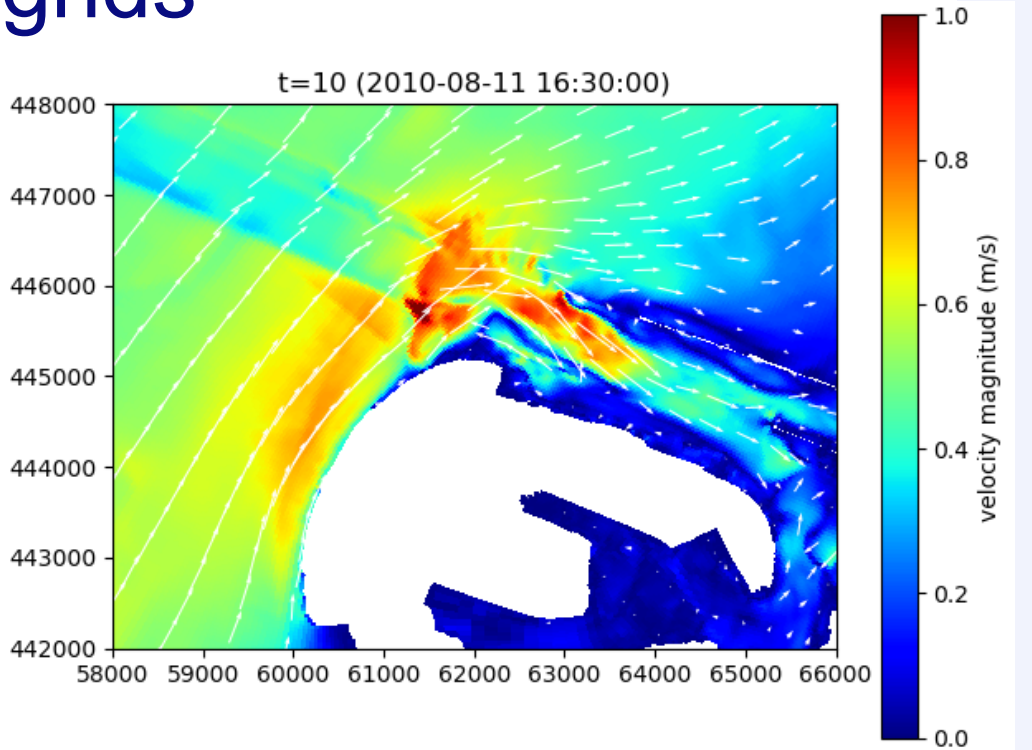


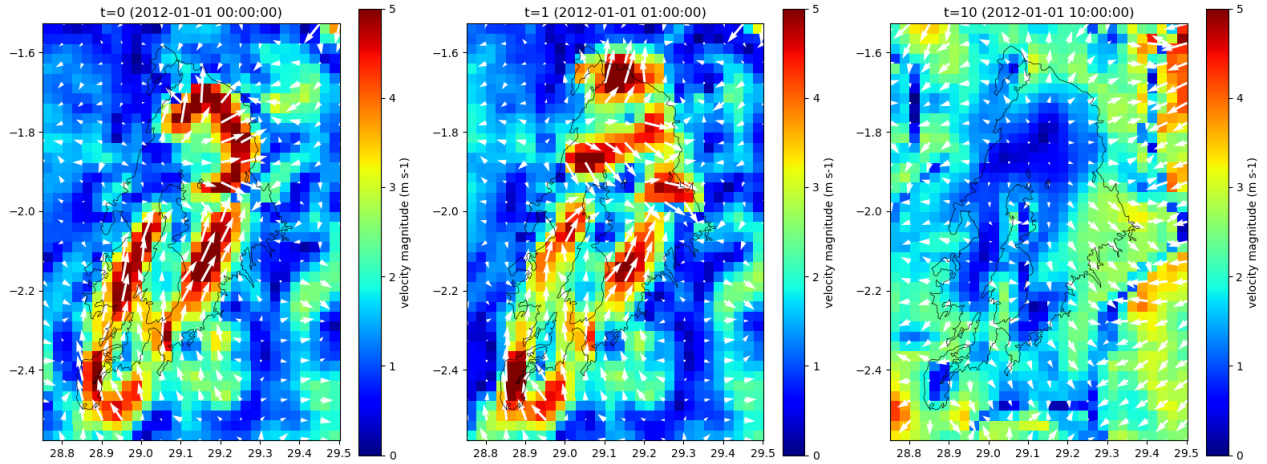


Examples: vector plots for regulargrids

- support for any other netCDF
- e.g.: SFINCS, Delft3D4, oceanmodels, meteomodels

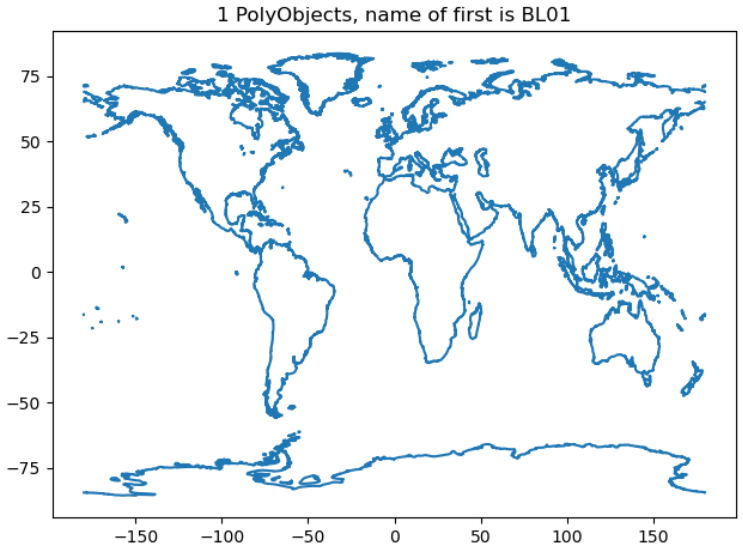


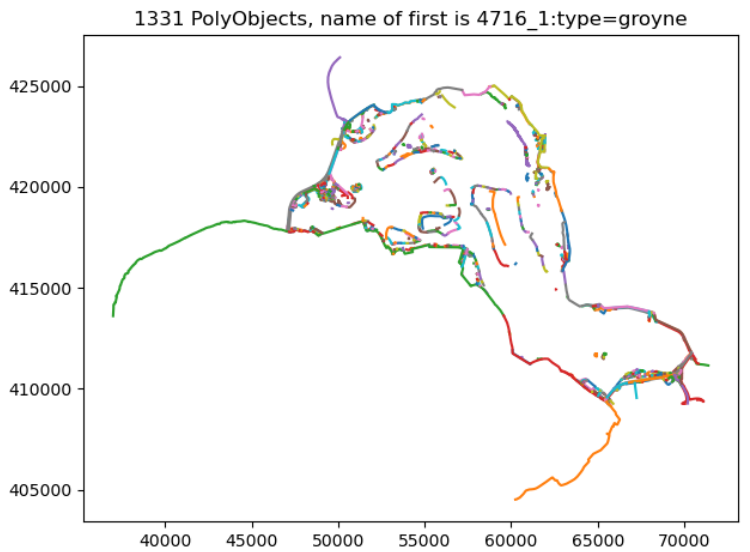


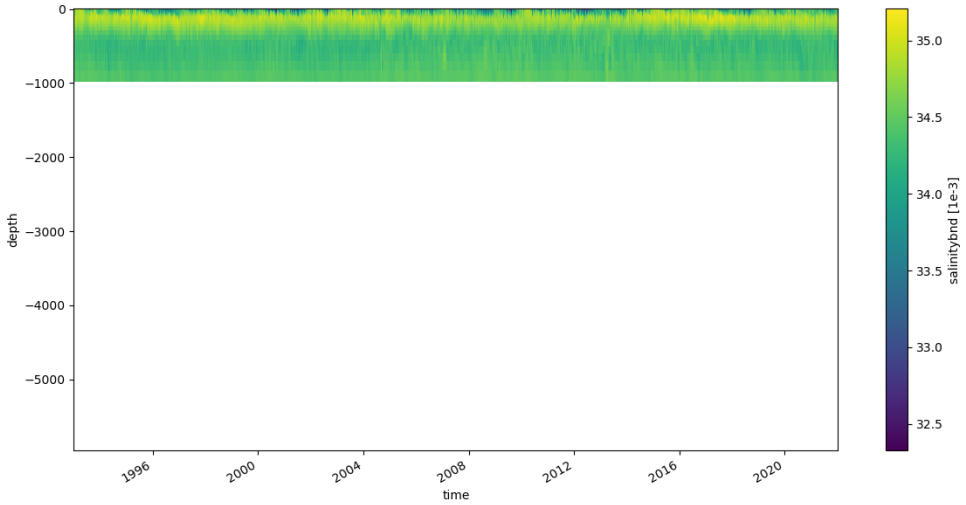


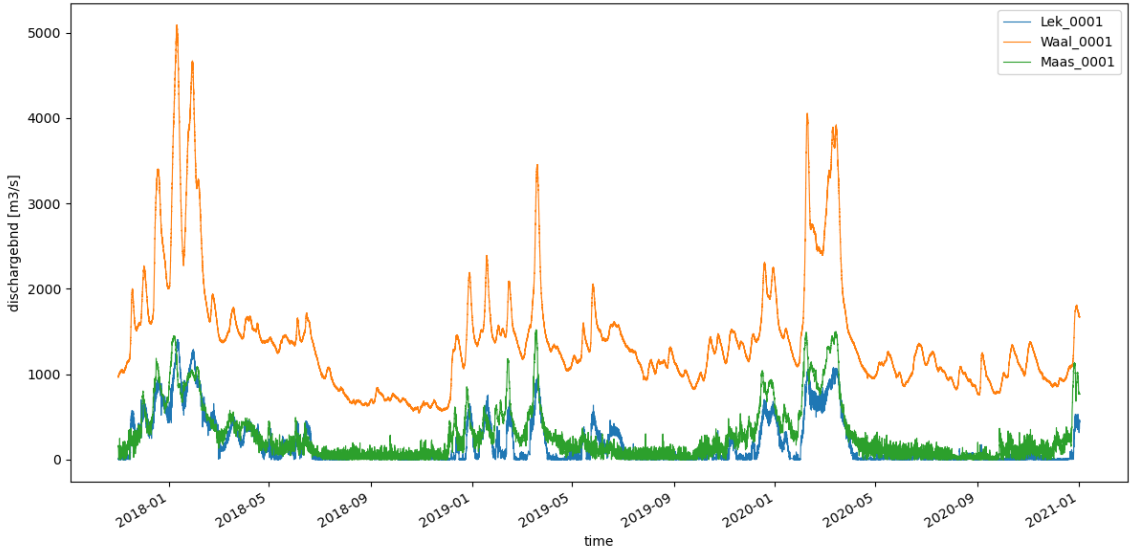
Examples: reading + plotting of D-FlowFM input files

- HYDROLIB-core for reading files
- conversion to xarray/pandas datasets with dfm_tools
- figures: pli/pliz/ldb/tek and bc files





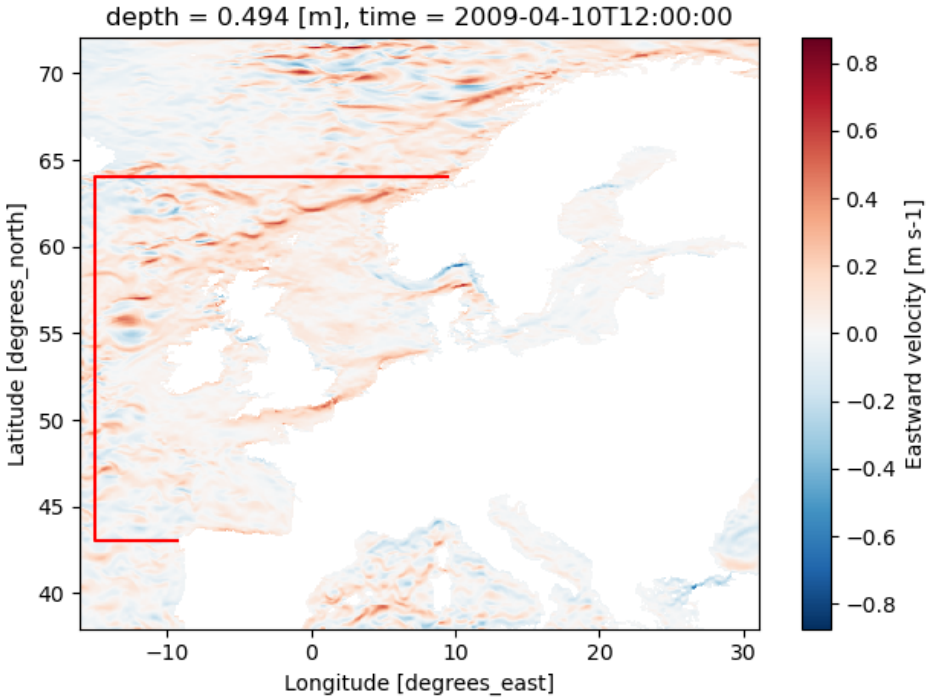


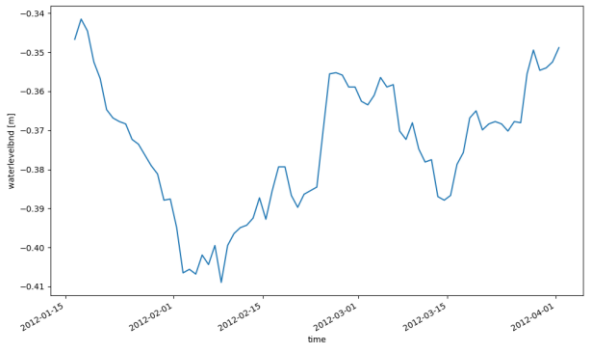


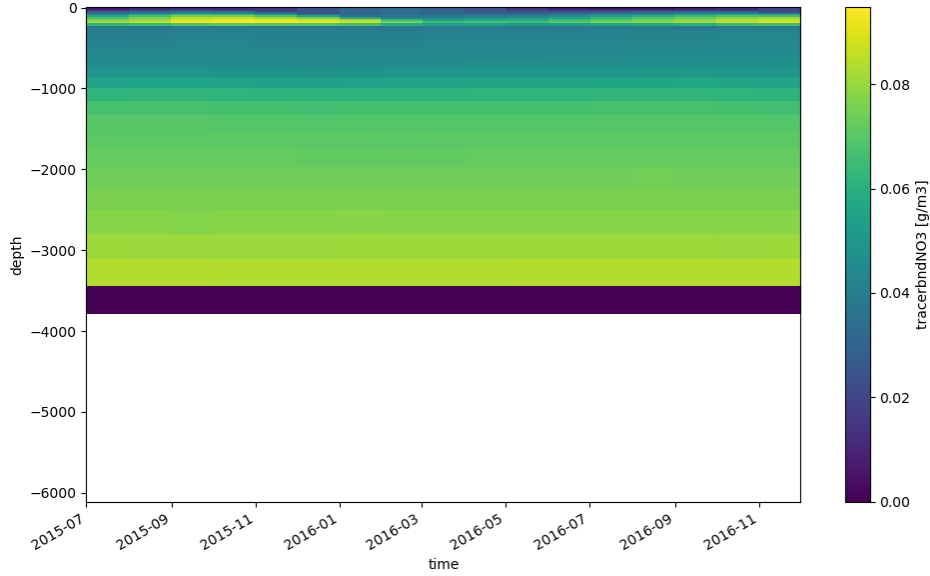
Deltares

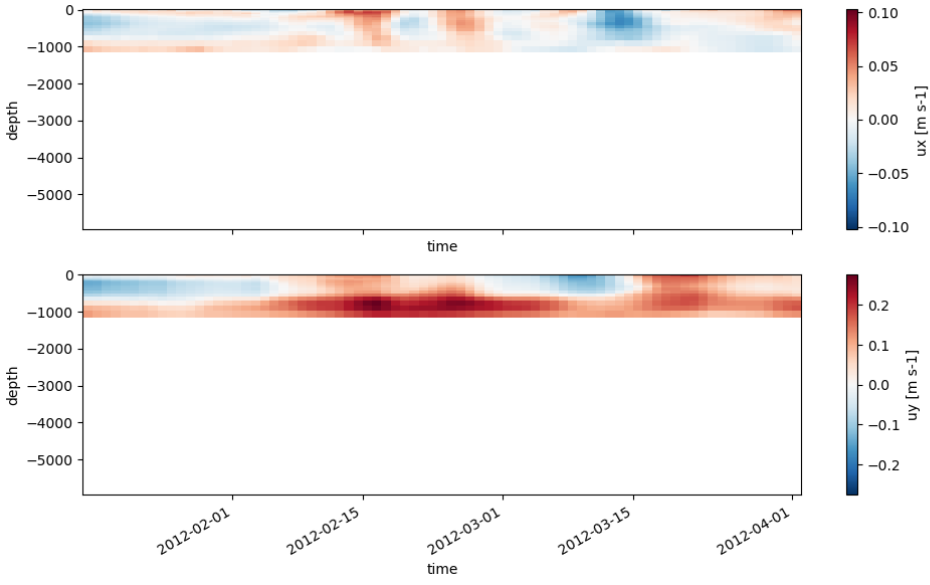
Examples: interpolate oceanmodel to bc

- xarray for reading netcdf data and .interp() to polyline points
- HYDROLIB-core for reading polyline and writing bc/ext files



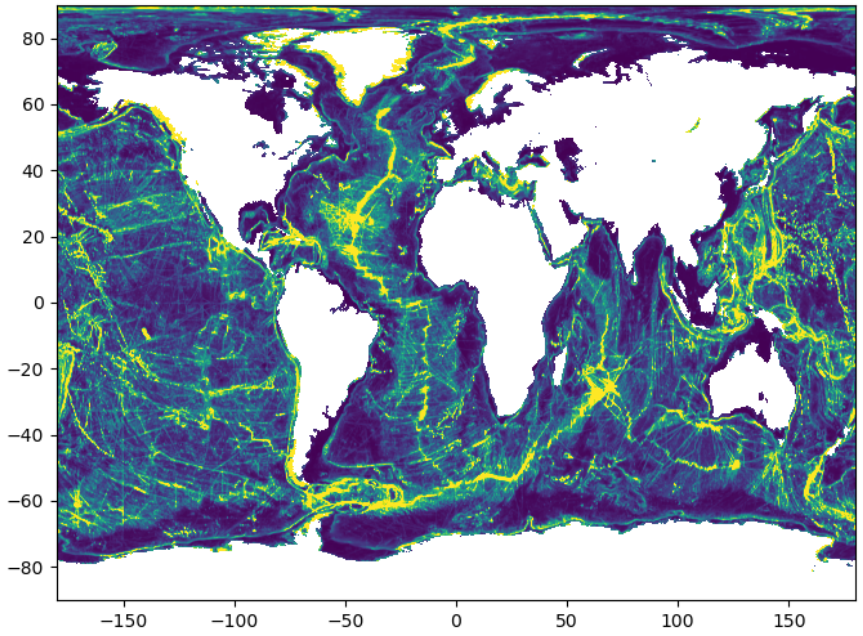


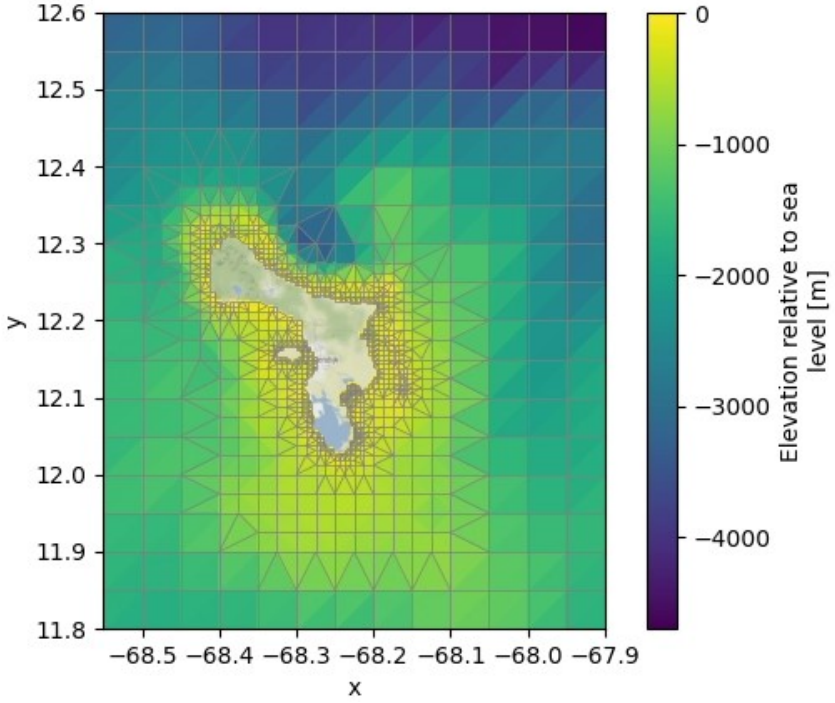




Recent developments

- grid generation + refinement based on bathymetry/waterdepth with **MeshkernelPy**
- usage of **xarray** in **dfm_tools**: merging+interpolation of model forcing
- development of 2D3D functionality in **HYDROLIB-core**
- BES hackaton: combining above into a **modelbuilder** “proof of concept”

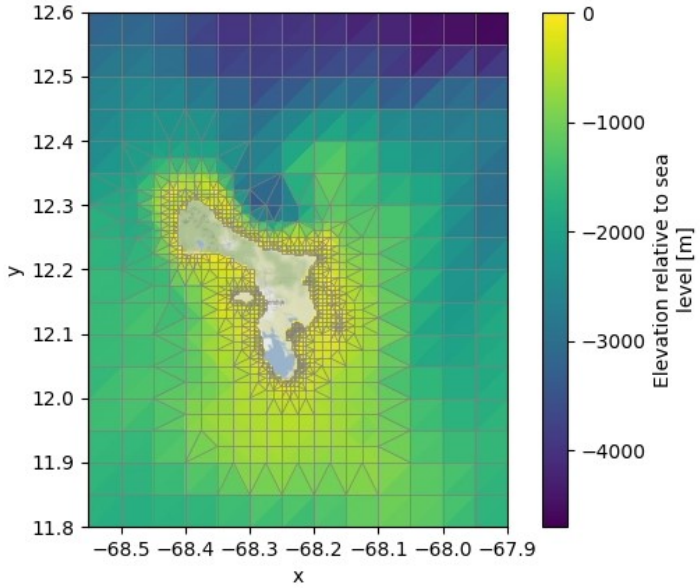


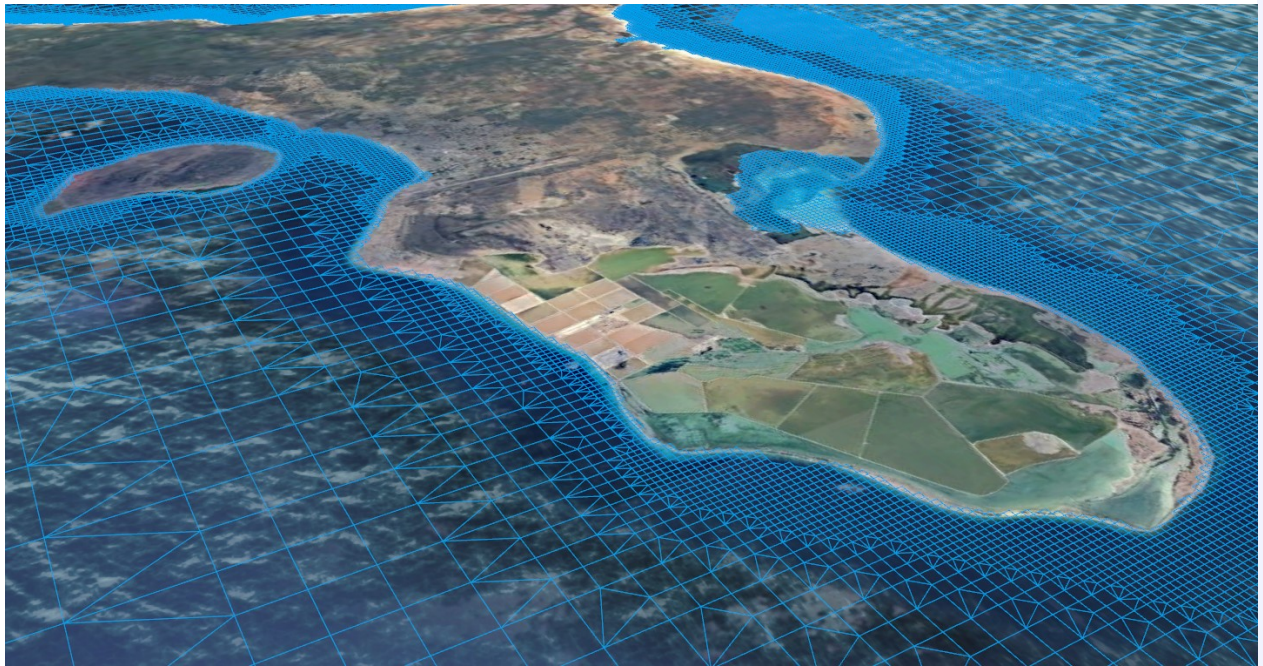


Combined into 3D Model Builder

Based on bounding box and period of interest, a 3D D-Flow FM model is created from global datasets:

- Using Python libraries **HYDROLIB-core, dfm_tools, MeshKernelPy, xarray** and others
- **Grid generation** and **refinement** (based on local water depth)
- **Bathymetry** (GEBCO) retrieval and interpolation to grid
- **Initial and open boundary conditions**: download and interpolate ocean currents, sea surface heights, salinity, temperature from CMEMS or others, tide from FES2014, EOT or GTSMv4.1
- **Meteorological forcing**: download ERA5





Deltares