

## **dfm\_tools**

**A Python package for D-FlowFM input  
and output**

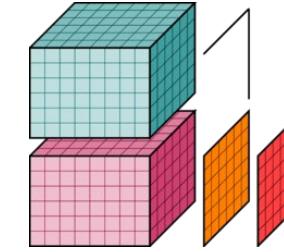
Jelmer Veenstra

# 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](#) and many more
- Why:
  - sharing of workflows among users (colleagues/external)
  - no need to reinvent the wheel
  - developments driven by projects like hydrolib-core, MassBay, FutureMares, TCOMS
- Where:
  - code on github: [https://github.com/Deltares/dfm\\_tools](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_00000000000000000000000000000000.nc'

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: float64 4.747e+0...
    station_y_coordinate: float64 4.188e+0...
    zcoordinate_c: float64 dask.array<chunksize=(145, 24, 36), meta=np.ndarray>
    zcoordinate_w: float64 dask.array<chunksize=(145, 24, 37), meta=np.ndarray>
    * time: datetime64[ns] 2007-...
    * stations: <U25 'GTS0-01' ....
    * source_sink: <U2 ''

Dimensions without coordinates: laydim, laydimw, source_sink_pts

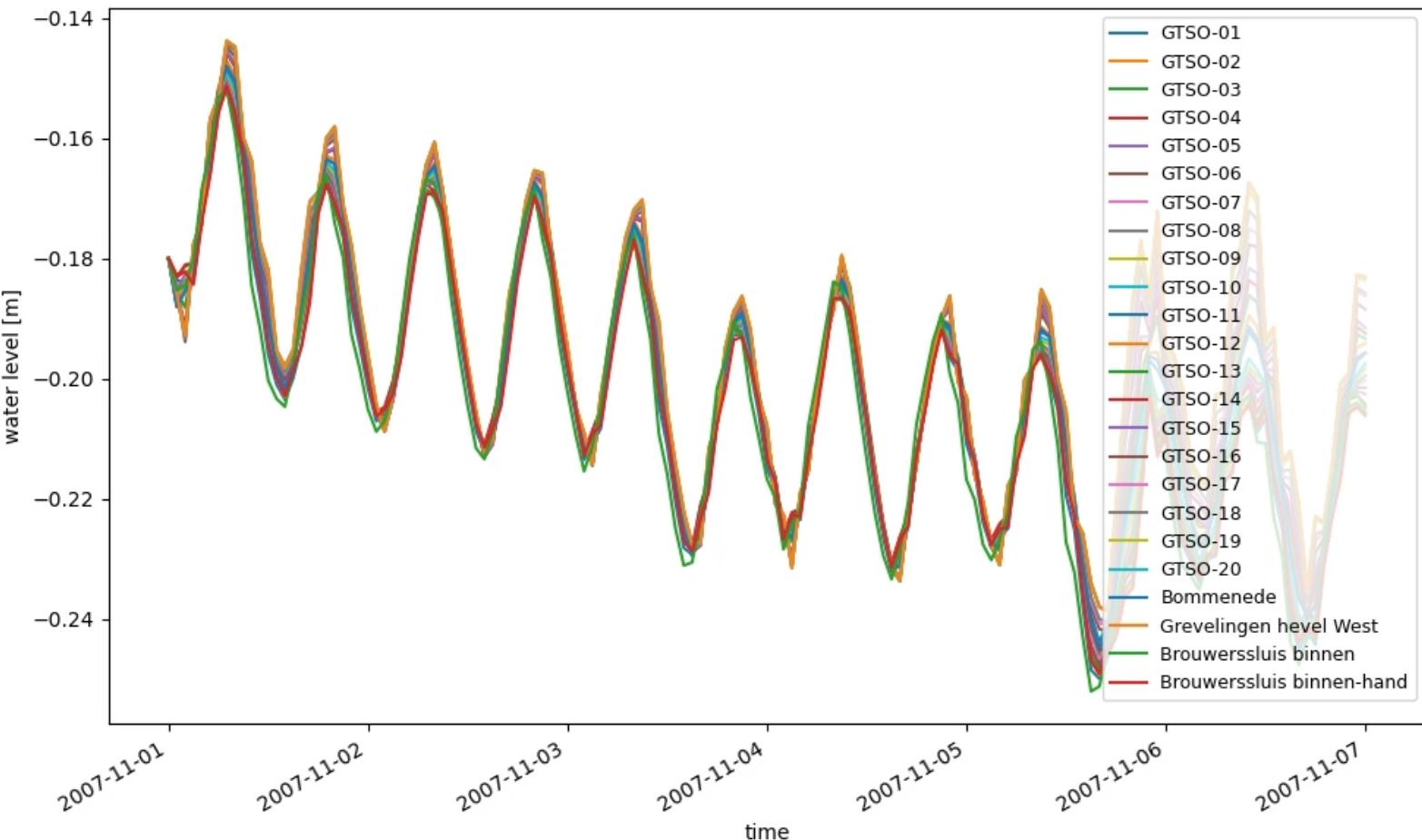
Data variables: (12/42)
    station_id: <S64 dask.array<chunksize=(24,), meta=np.ndarray>
    waterlevel: float64 dask.array<chunksize=(145, 24), meta=np.ndarray>
    bedlevel: float64 dask.array<chunksize=(24,), meta=np.ndarray>
    waterdepth: float64 dask.array<chunksize=(145, 24), meta=np.ndarray>
    x_velocity: float64 dask.array<chunksize=(145, 24, 36), meta=np.ndarray>
    y_velocity: float64 dask.array<chunksize=(145, 24, 36), meta=np.ndarray>
    ...
```

# xarray: io of netcdf files



- easy indexing (subsetting)
- easy and complete plotting
- .../tests/examples/postprocess\_hi  
snc.py

```
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')
```



# HYDROLIB: io of D-FlowFM input files



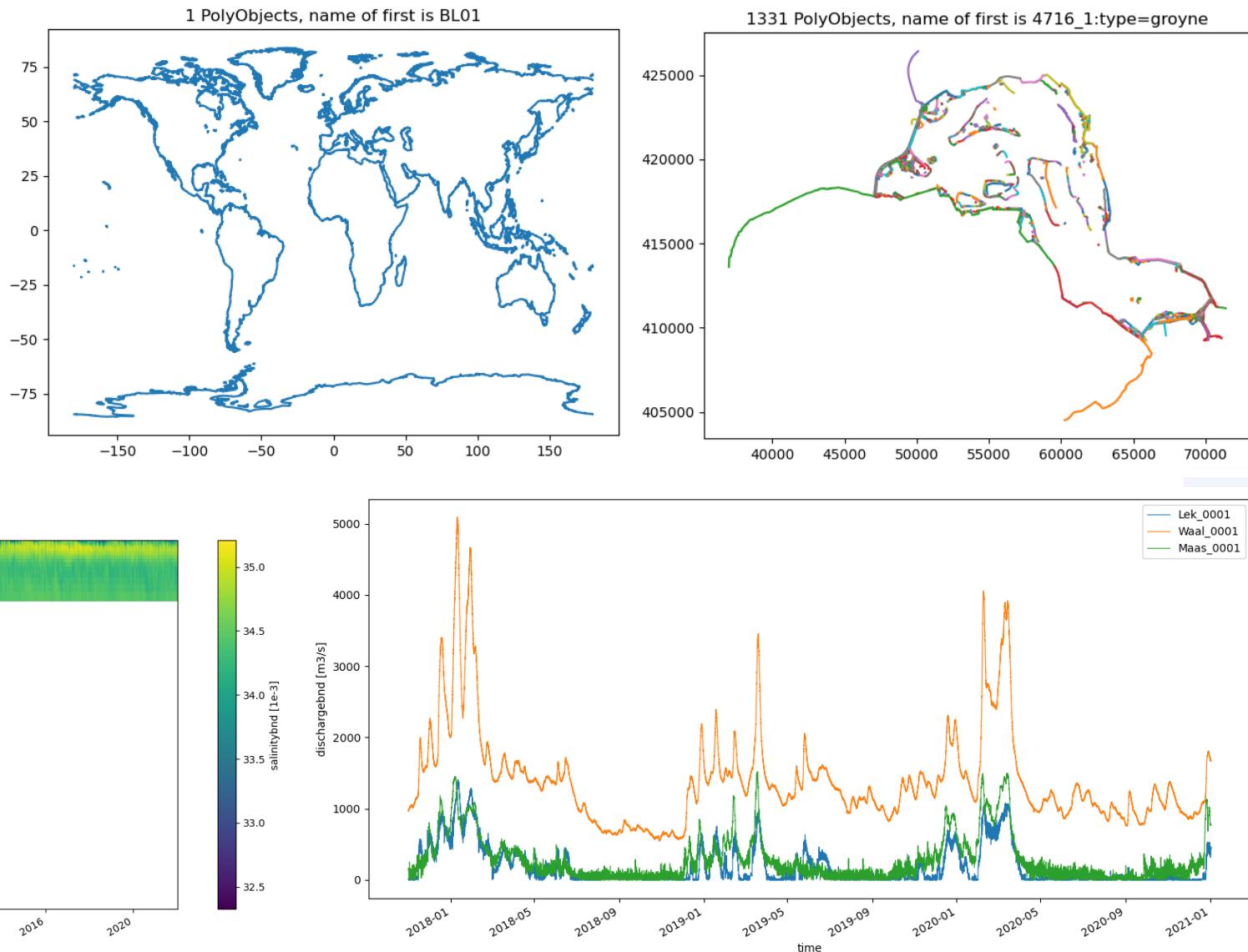
Core scripts & tools for interaction with D-FlowFM:

- reading/writing of (almost all) D-FlowFM input files (mdu, ext, bc, pli, etc)
- full functionalities:  
[https://deltares.github.io/HYDROLIB-core/0.3.1/topics/dhydro\\_support/](https://deltares.github.io/HYDROLIB-core/0.3.1/topics/dhydro_support/)
- supported, documented, maintained
- available on github:  
<https://github.com/Deltares/HYDROLIB-core>
- users do not need to re-invent the wheel about “basic stuff/file formats”, focus on the analysis itself



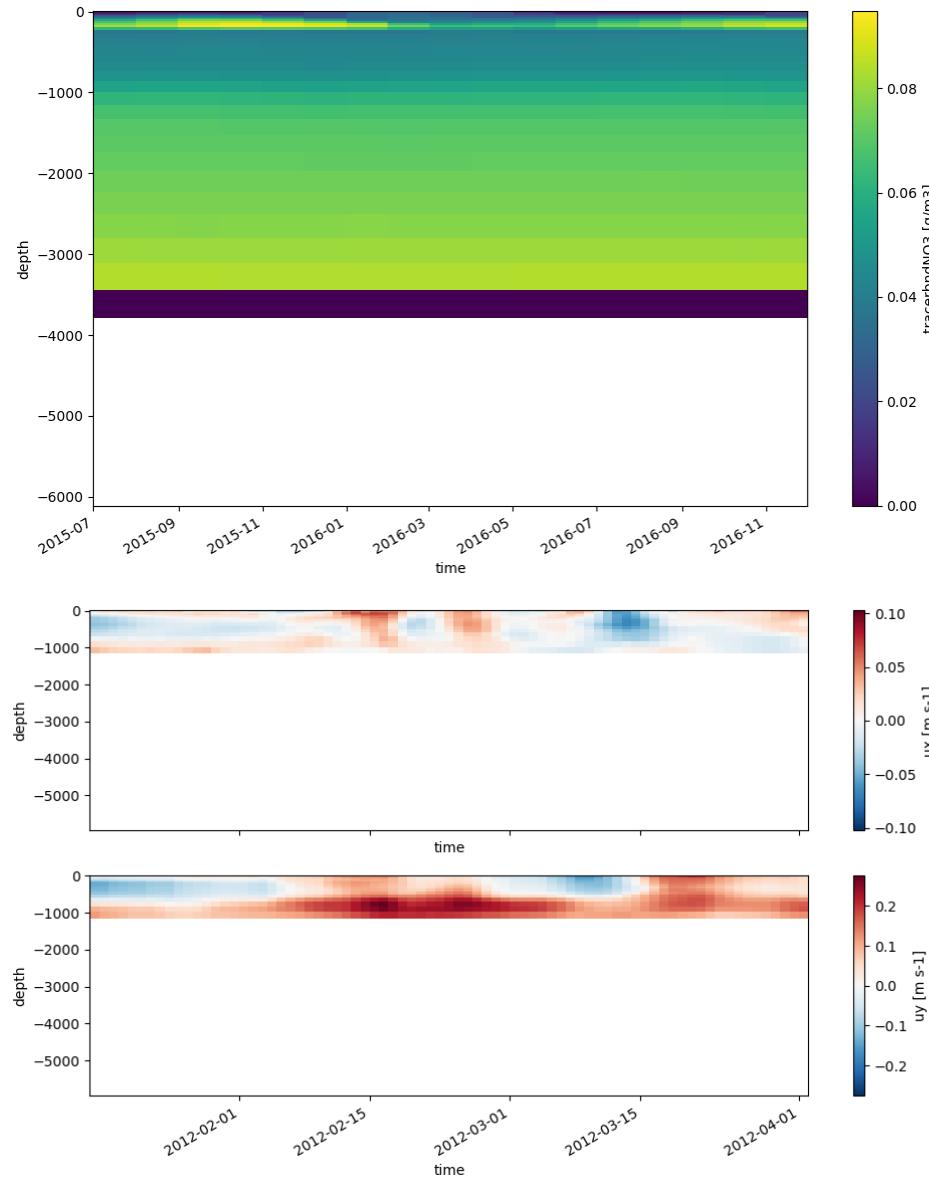
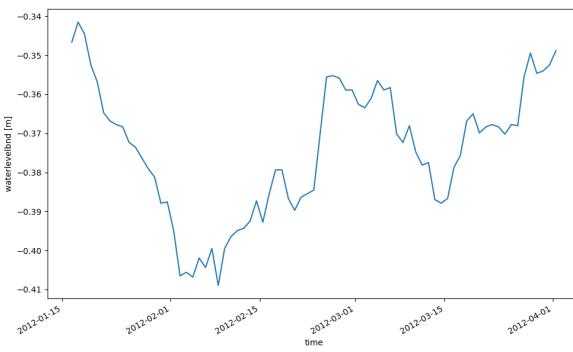
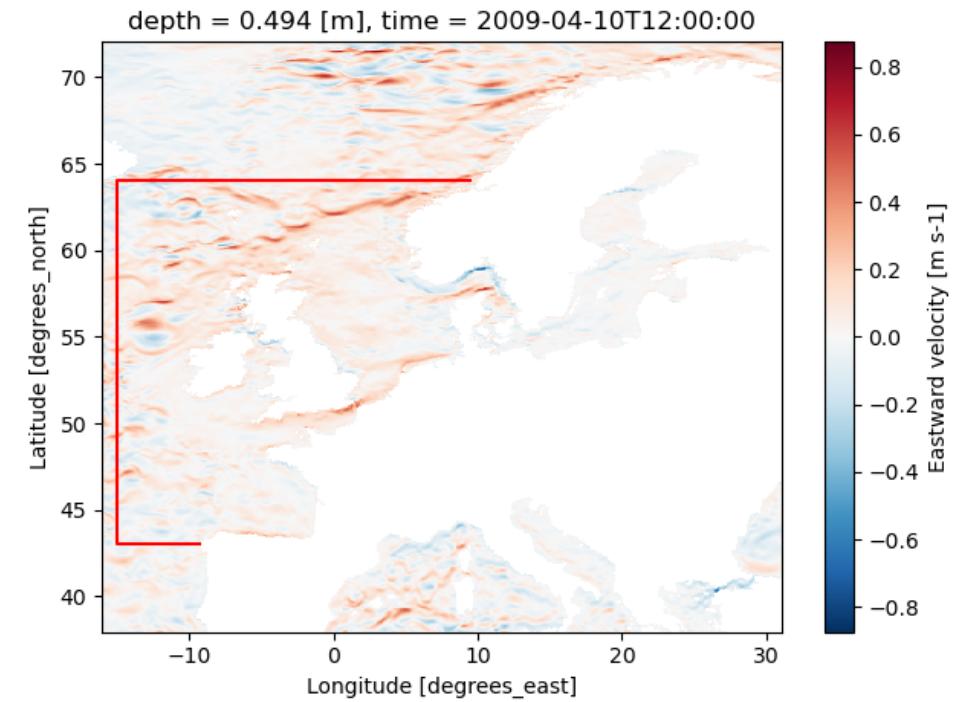
# Examples: reading+plotting of D-FlowFM input files

- hydrolib for reading files
- conversion to xarray/pandas dataset with dfm\_tools
- figures: pli/pliz/ldb/tek and bc files
- `.../tests/examples/preprocess_hydrolib_readbc.py`
- `.../tests/examples/preprocess_hydrolib_readwritepol.py`



# Examples: interpolate oceanmodel to bc

- xarray for reading netcdf data and `.interp()` to polyline points
- hydrolib for reading pli and writing bc/ext files
- `.../tests/examples/preprocess_interpolate_nc_to_bc.py`

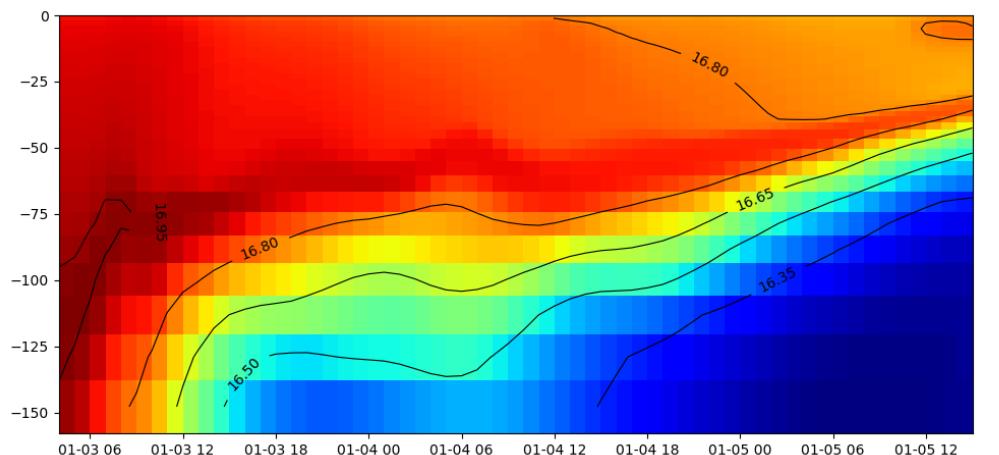
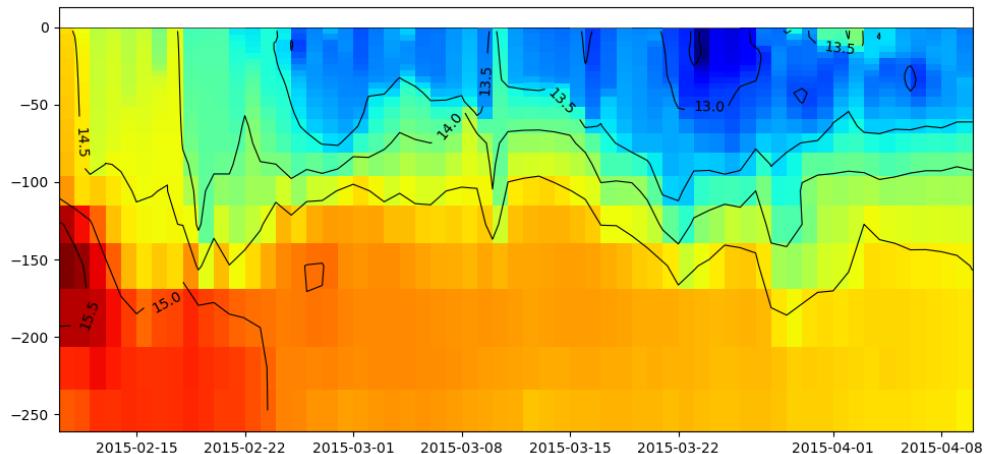
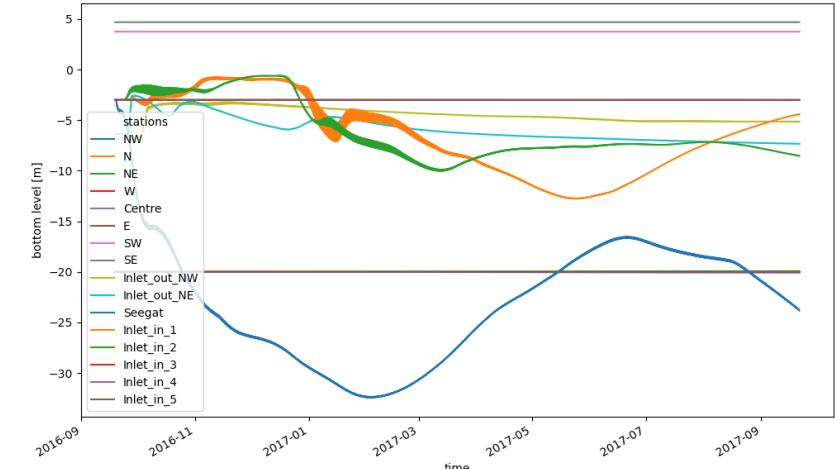


# Application for regional models

- Manually defined model boundaries, used for:
  - Grid cut-out of GTSM (currently manual)
  - Download of ERA5 and CMEMS field
    - `.../tests/examples/download_meteodata_oceandata.py`
  - Generation of (2D) open boundary forcing from FES2014 (tidal constituents)
    - `.../tests/examples/preprocess_interpolate_nc_to_bc.py`
  - Generation of (3D) open boundary forcing from CMEMS (salinity, temperature and sealevel anomaly)
    - `.../tests/examples/preprocess_interpolate_nc_to_bc.py`
  - Generation of 3D initial fields from CMEMS (salinity and temperature)
    - `.../tests/examples/preprocess_ini_cmems_to_nc.py`
  - Generation of meteorological forcing fields
    - `.../tests/examples/preprocess_meteo_mergenetcDFtime_xarray.py`
- The *dfm\_tools* enables the quick set-up of the basis of a 3D model

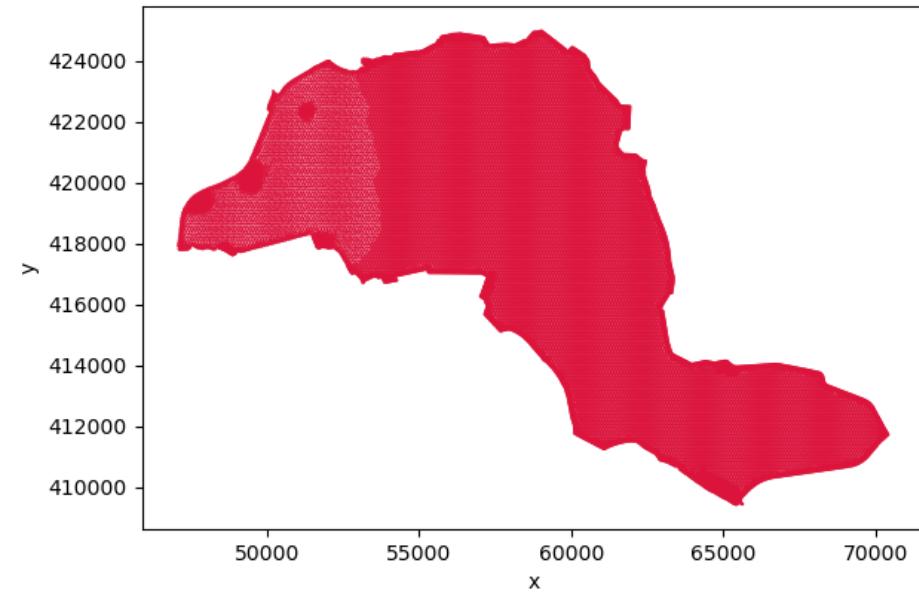
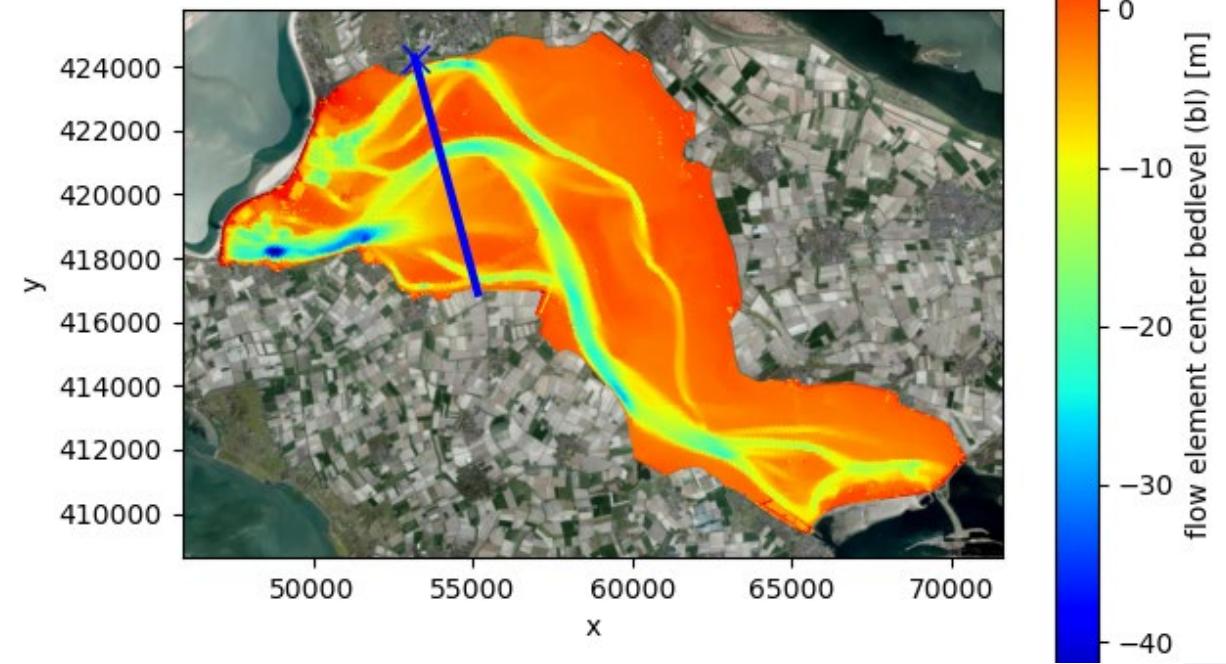
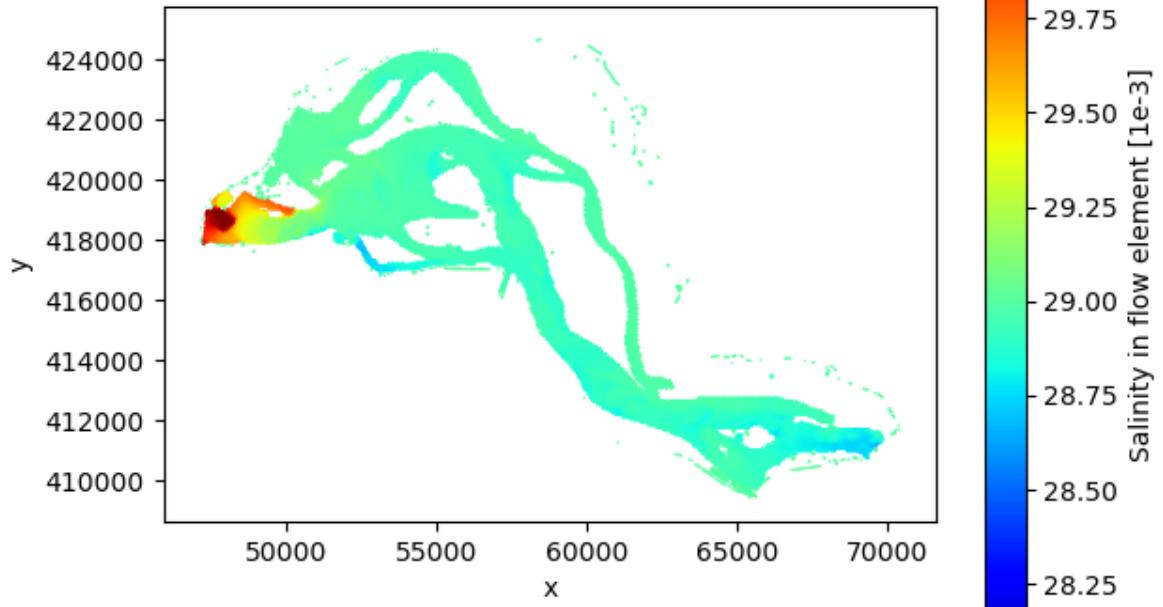
# Examples: read+plot hisfile

- xarray for reading hisfile (D-FlowFM and sfincs)
- easy retrieval of lists of stations, cross\_sections, general structures, etc thanks to preprocess\_hisnc()
- selection of data over dimensions: time, depth, stations, cross\_sections, structures, etc
- zt-plot (timeseries of one point with depth dim)
- .../tests/examples/postprocess\_hisnc.py
- coming up:
  - interpolation
  - nesting



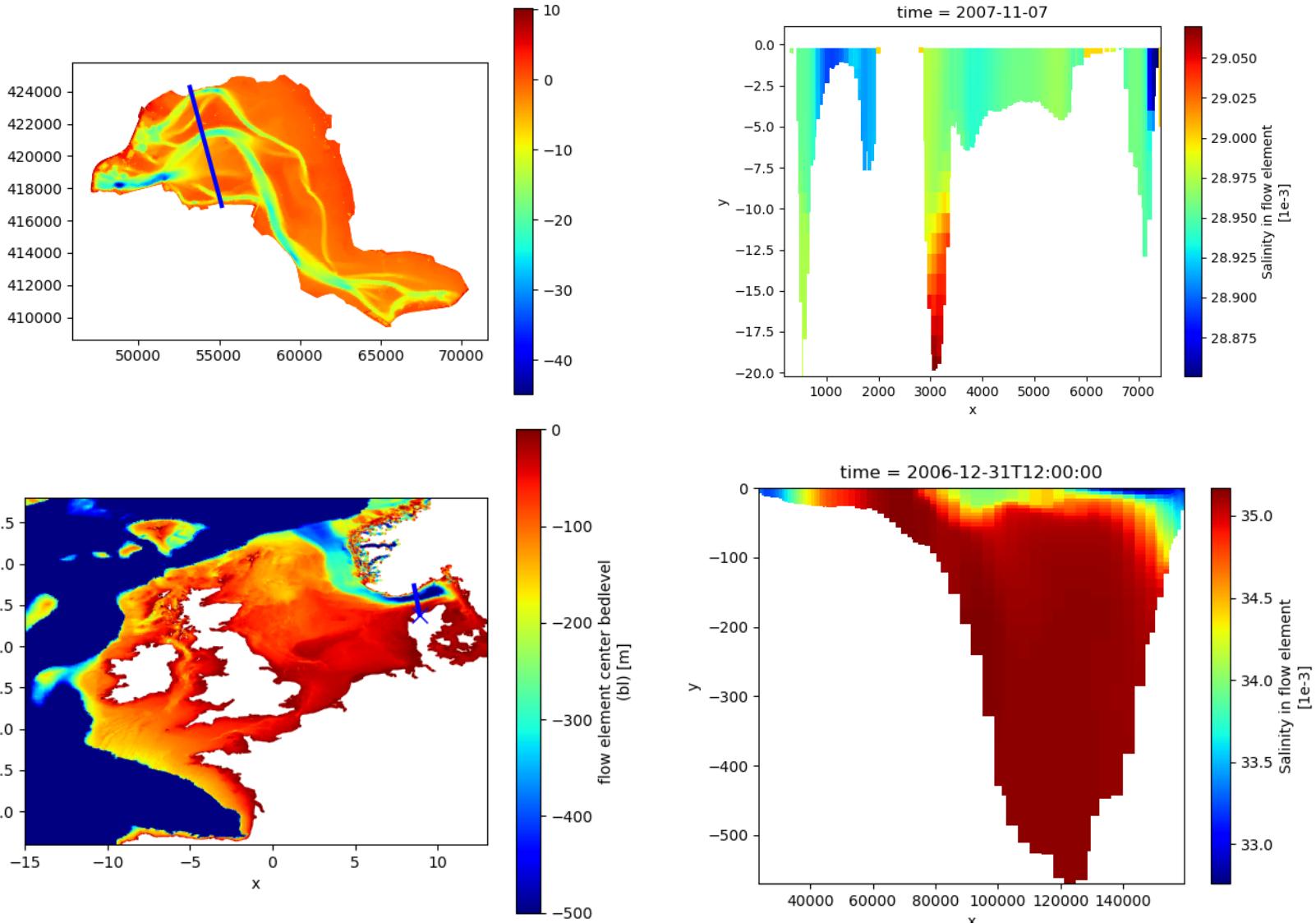
# Examples: read+plot mapfiles

- 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
- 3D slicing: next slide



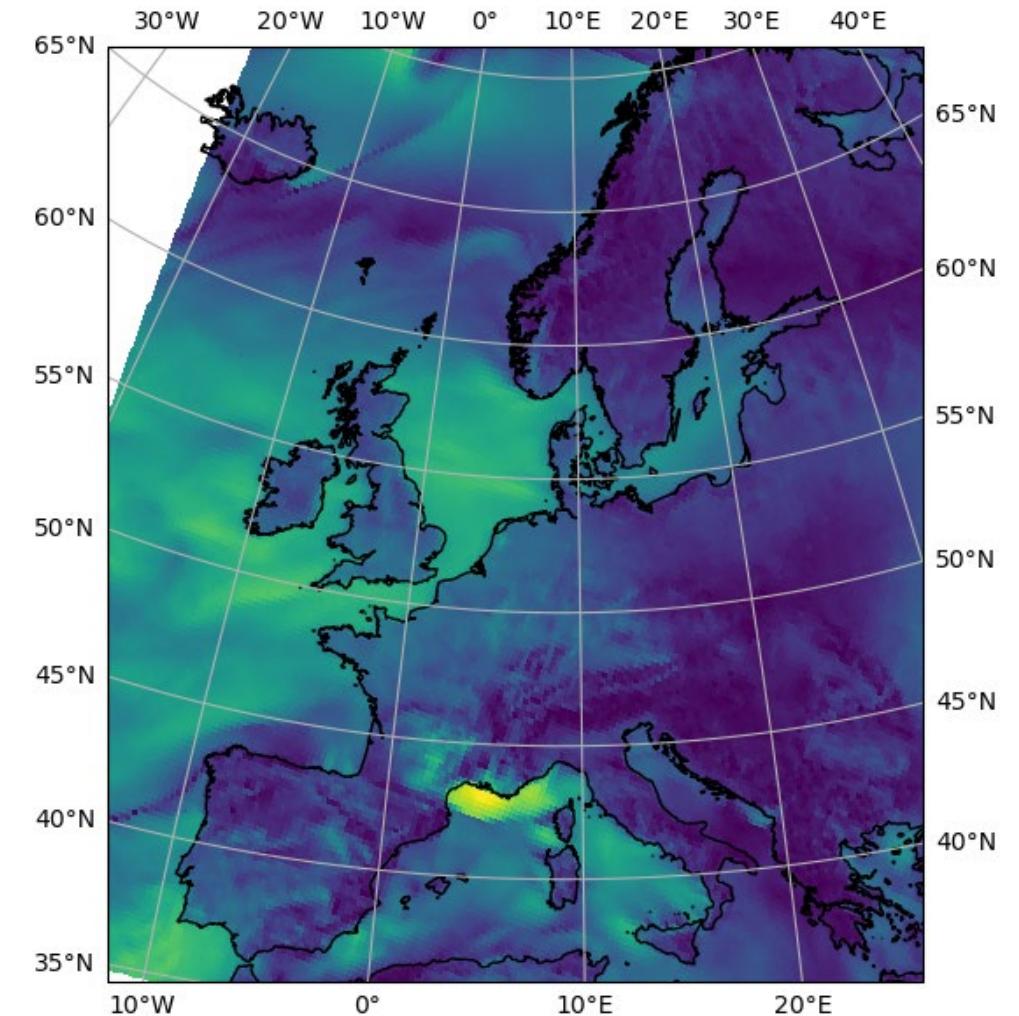
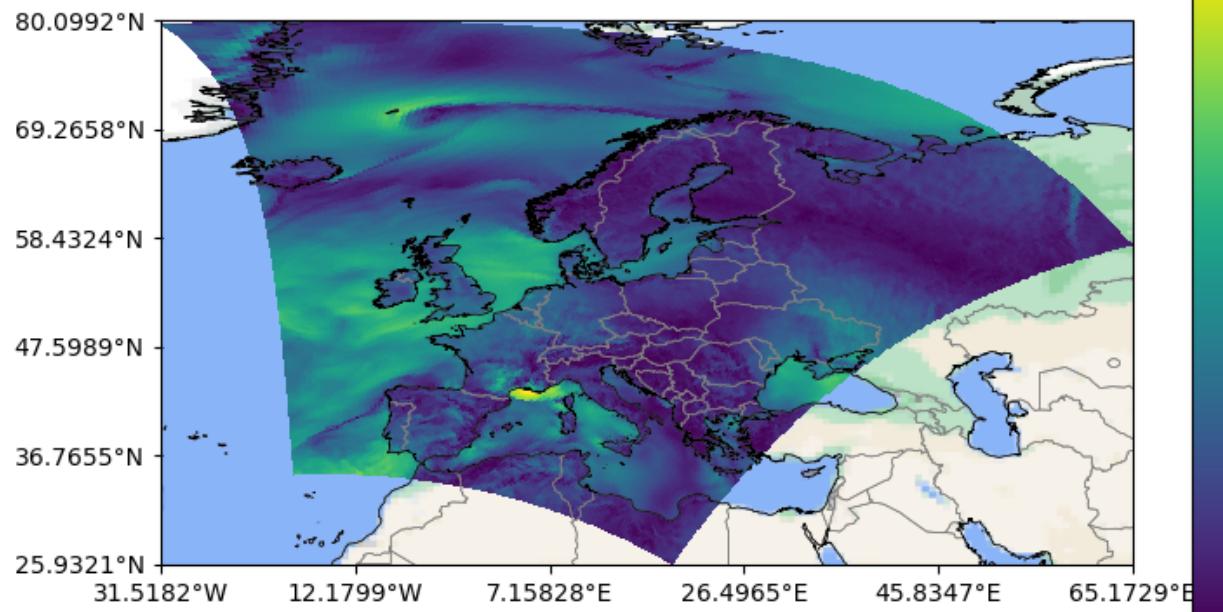
# Examples: slice trough 3D partitioned FM map data

- given a polyline (blue)
- slice trough 3D partitioned FM map data
- returns and xugrid DataArray so plotting is easy
- also possible to select values based on depth w.r.t.  $z_0/wl/bl$



# 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

