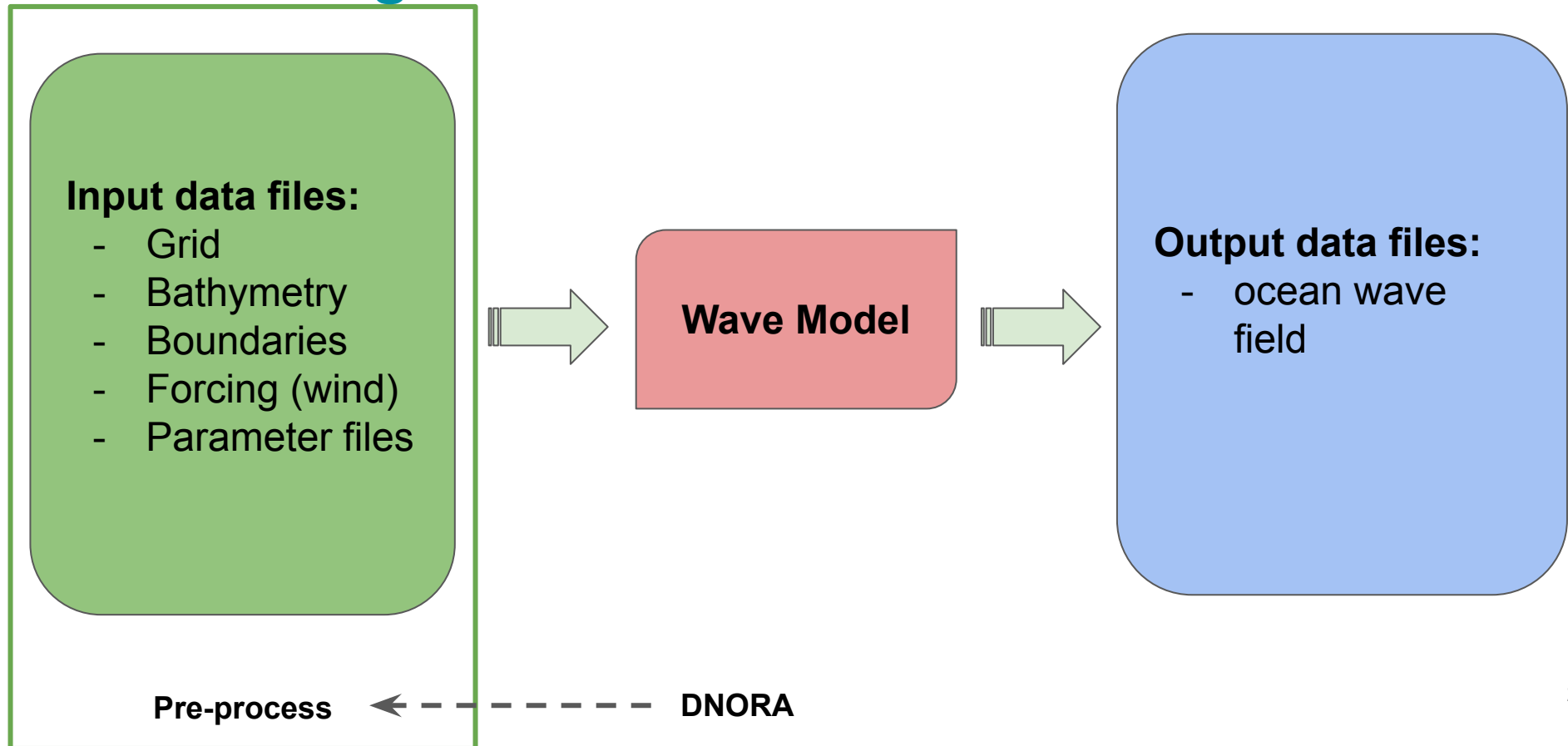


DNORA: Open-access dynamical downscaling of open-ocean wave hindcast/forecast for coastal areas

Konstantinos Christakos (IMT/NTNU & MET Norway) &
Jan-Victor Björkqvist (MET Norway & FMI)

5th Coastal Engineering Day, NTNU/PIANC, 31.03.2022

Running a wave model



What is DNORA?

DNORA is a Python package for **D**ynamical downscaling of **NORA** wave hindcast using the wave models SWAN, WAVEWATCH III, SWASH, and HOS-ocean.

The package contains functions that:

- Create a high-resolution grid using open-access bathymetry/topography datasets
- Prepare the boundary conditions (NORA3, WAM4)
- Prepare the wind (NORA3, WAM4) forcing
- Create input parameter files for the wave models (SWAN, SWASH, HOS-ocean)
- Run the model (SWAN, SWASH, HOS-ocean)

Available at <https://github.com/KonstantinChri/dnora>

The screenshot displays the GitHub repository page for **KonstantinChri / dnora**. The repository is public and has 1 star and 2 forks. The main content area shows a file browser with the following commit history table:

File	Commit Message	Author	Time
dnora	Bug fix in spectral processor	Jan-Victor Björkqvist	last month
docs	Merge branch 'develop' into main		3 months ago
examples	Added EmptyTopo possiility in one example		3 months ago
tests/unittests	Bug fix in spectral processor		last month
.gitignore	Updated gitignore to include pycache for new modules		3 months ago
LICENSE	Added licence and readme.		4 months ago
README.md	remove old example		3 months ago
environment.yml	add environment.yml		3 months ago

The README preview shows the **dnora** logo and the text: "What is dnora? dnora is a software for dynamical downscaling of wave products i.e., NORA3 wave hindcast and WAM4 wave forecast from Norwegian Meteorological Institute. More information in [documentation](#)".

The sidebar on the right contains the following information:

- About:** Dynamical downscaling of NORA3 wave hindcast. Includes Readme, GPL-2.0 License, 1 star, 1 watching, and 2 forks.
- Releases:** 5 releases. Latest release is **Release v1.1.3** on 18 Feb.
- Packages:** No packages published. Publish your first package.
- Contributors:** 2 contributors: KonstantinChri (Konstantinos Christakos) and bjorkqvi (Jan-Victor Björkqvist).
- Languages:** Python 99.9%, Shell 0.1%.

Ideology of DNORA

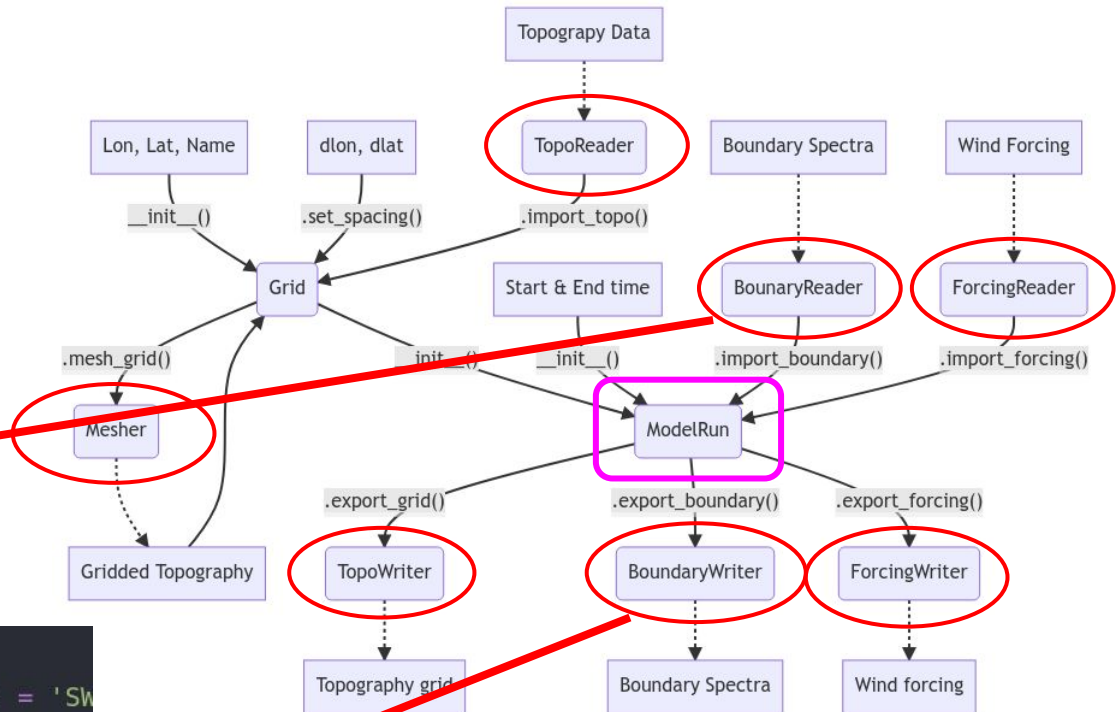
```
class NORA3(BoundaryReader):
    def __init__(self, stride: int=24, hours_per_file: int=2):
        self.stride = copy(stride)
        self.hours_per_file = copy(hours_per_file)
        self.lead_time = copy(lead_time)
        self.last_file = copy(last_file)

    return

    def convention(self) -> str:
        return 'Ocean'
```

```
class SWAN(BoundaryWriter):
    def __init__(self, factor = 1E-4, out_format = 'SWAN'):
        self.factor = factor
        self.out_format = out_format
        return

    def _convention_in(self) -> str:
        """Convention of spectra"""
        return 'Met'
```

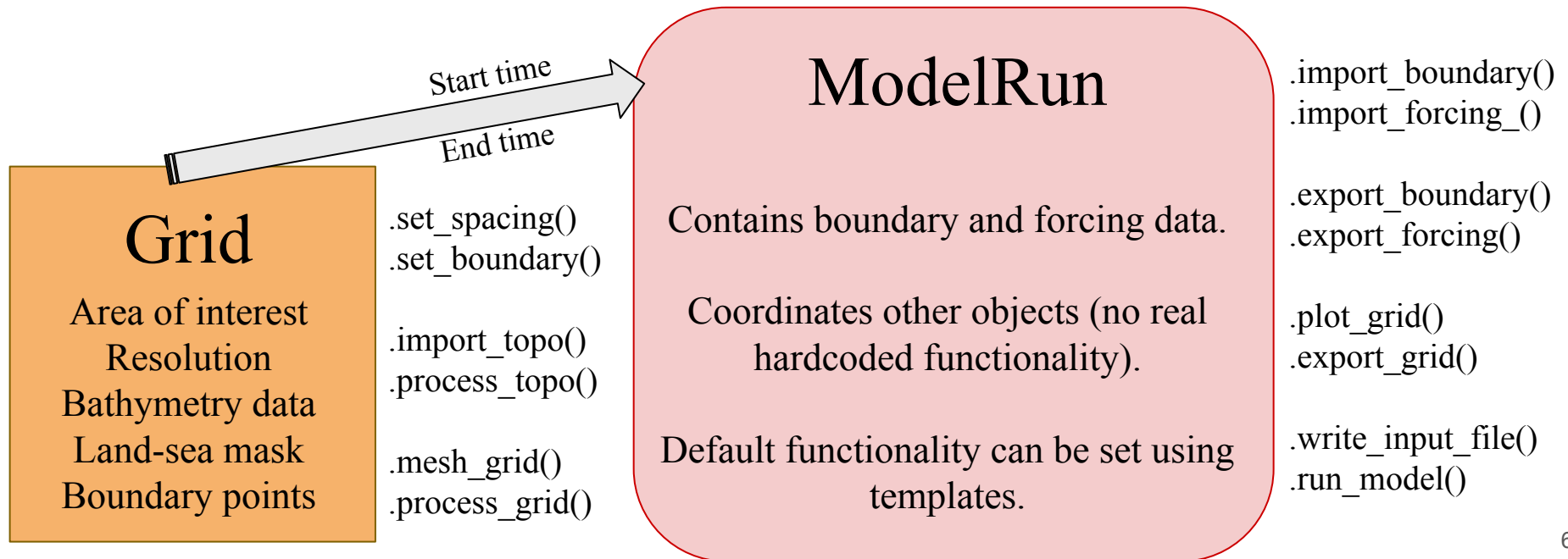


Functionality can be **added incrementally** based on needs.

Typical combinations can also be **packaged as templates** for quick and easy use.

Structure of DNORA

To use DNORA the user needs to work directly with two objects:

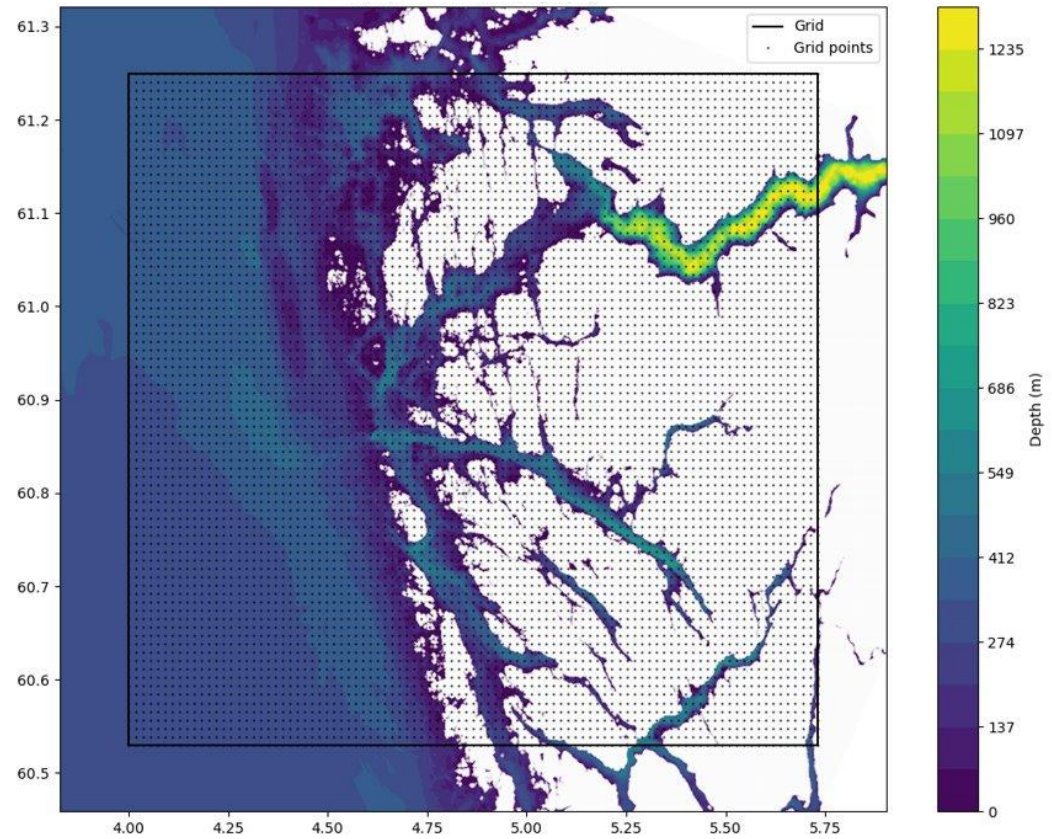


Defining regular grid

```
mdl.plot_grid()
```

Input data files:

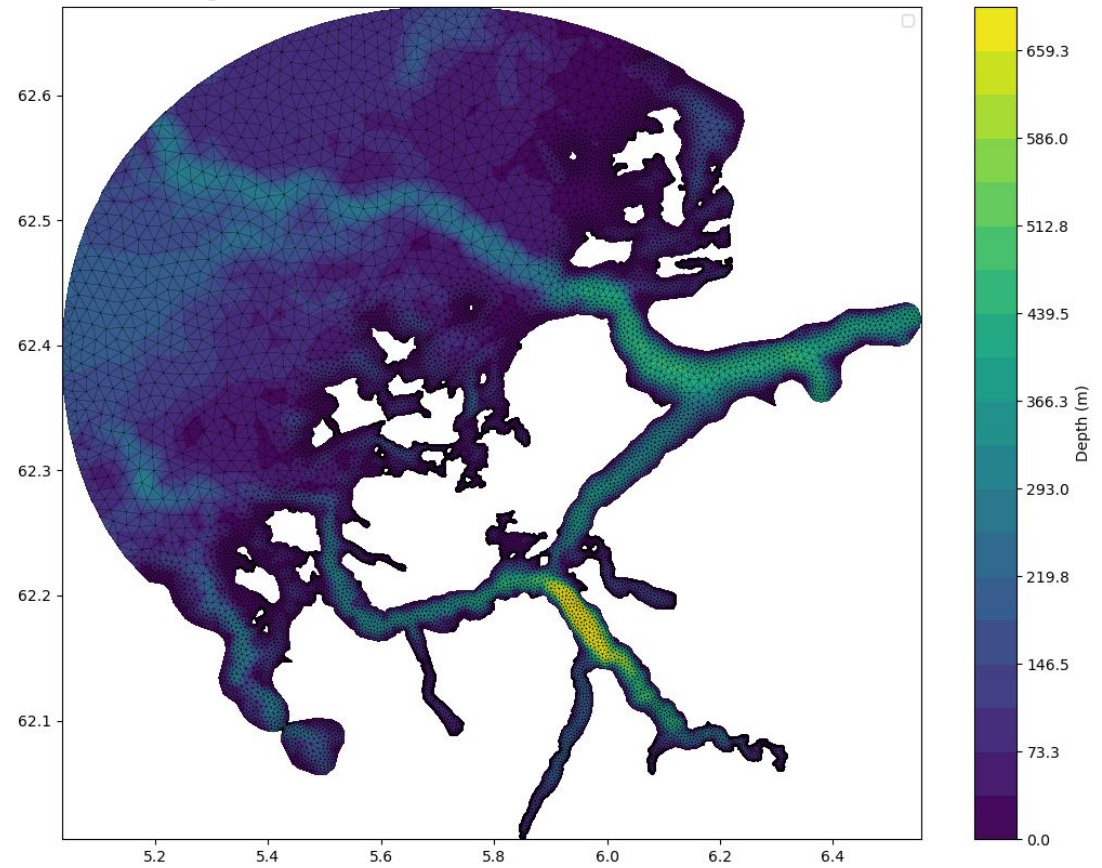
- Grid
- Bathymetry



Defining unstructured grid

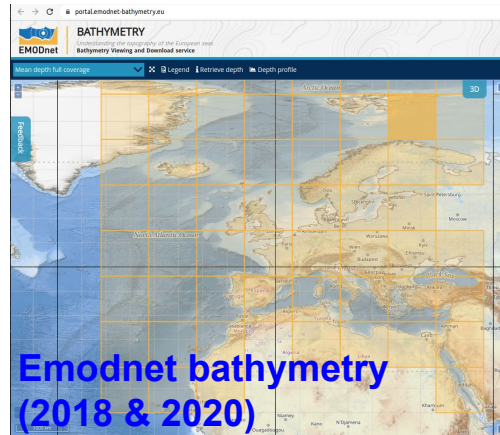
```
mdl.plot_grid()
```

Input data files:
- Grid
- Bathymetry

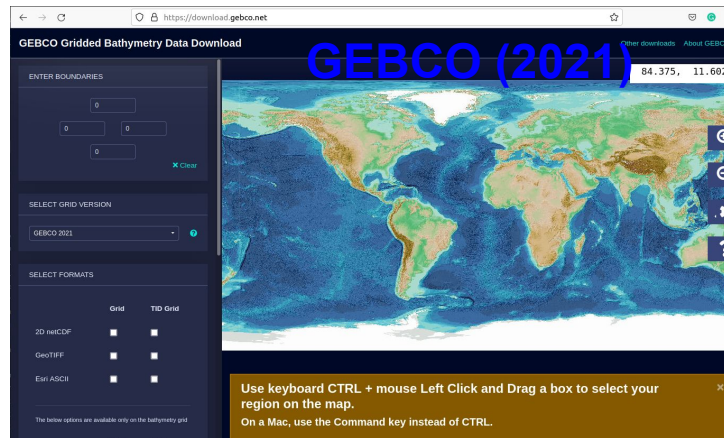
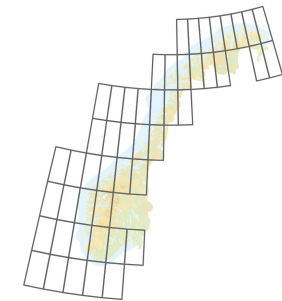


Input data files:

- Grid
- Bathymetry



Kartverket50mNo

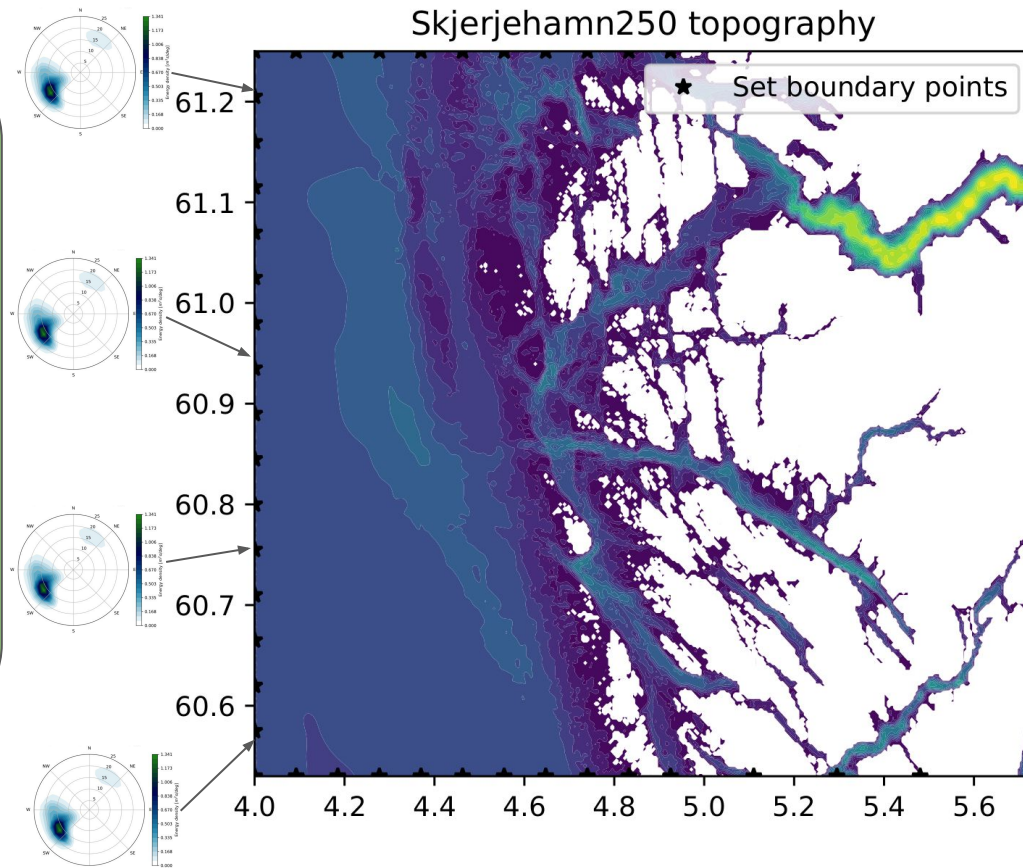


Defining Boundaries

```
mdl.plot_grid()
```

Input data files:

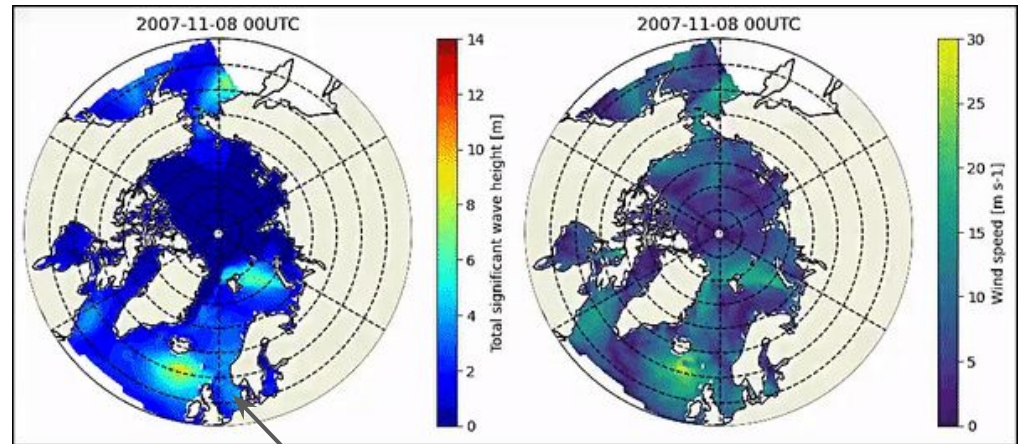
- Grid
- Bathymetry
- **Boundaries**



Boundaries & Forcing (NORA3:1992-2020)

Input data files:

- Grid
- Bathymetry
- **Boundaries**
- **Forcing**



NORA3
lon.=3.0574315,lat=56.56322
2007-11-08T00
 $\theta_p=172.5\text{deg}$
 $T_p=12.28\text{s}$

← → ↻ thredds.met.no/thredds/projects/windsurfer.html

Norwegian Meteorological Institute **Catalog <https://thredds.met.no/thredds/projects/windsurfer.html>**

Terms of service

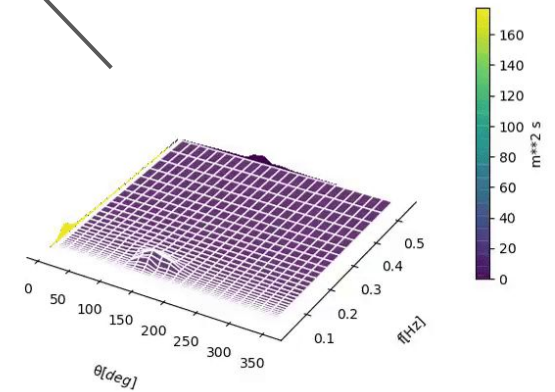
This is a shared public service which may experience overload in traffic from time to time. We reserve the right to block IP addresses if they cause traffic overload. E to answer before the next request is sent. If you require priority access for your own operational service please contact us at thredds@met.no. Get messages about when the problem occurred.

[MET Norway's Privacy Policy](#)

Dataset

- Windsurfer MyWawWam 3km hindcast
- MyWaveWam 3km Grid Files/
- MyWaveWam 3km Grid Aggregated/
- MyWaveWam 3km Spectra Files/

[MET Norway Thredds Service at Norwegian Meteorological Institute](#) see [Info](#) [Documentation](#)

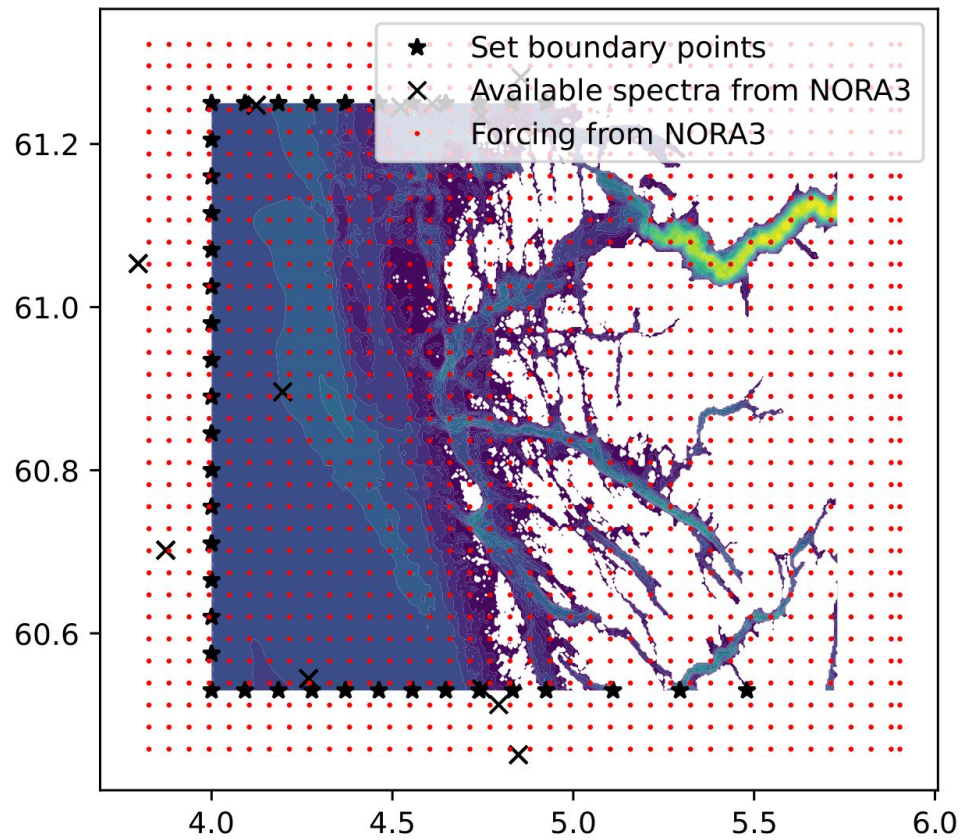


Defining boundaries and forcing

```
mdl.plot_grid()
```

Skjerjehamn250 topography

- Input data files:**
- Grid
 - Bathymetry
 - **Boundaries**
 - **Forcing**

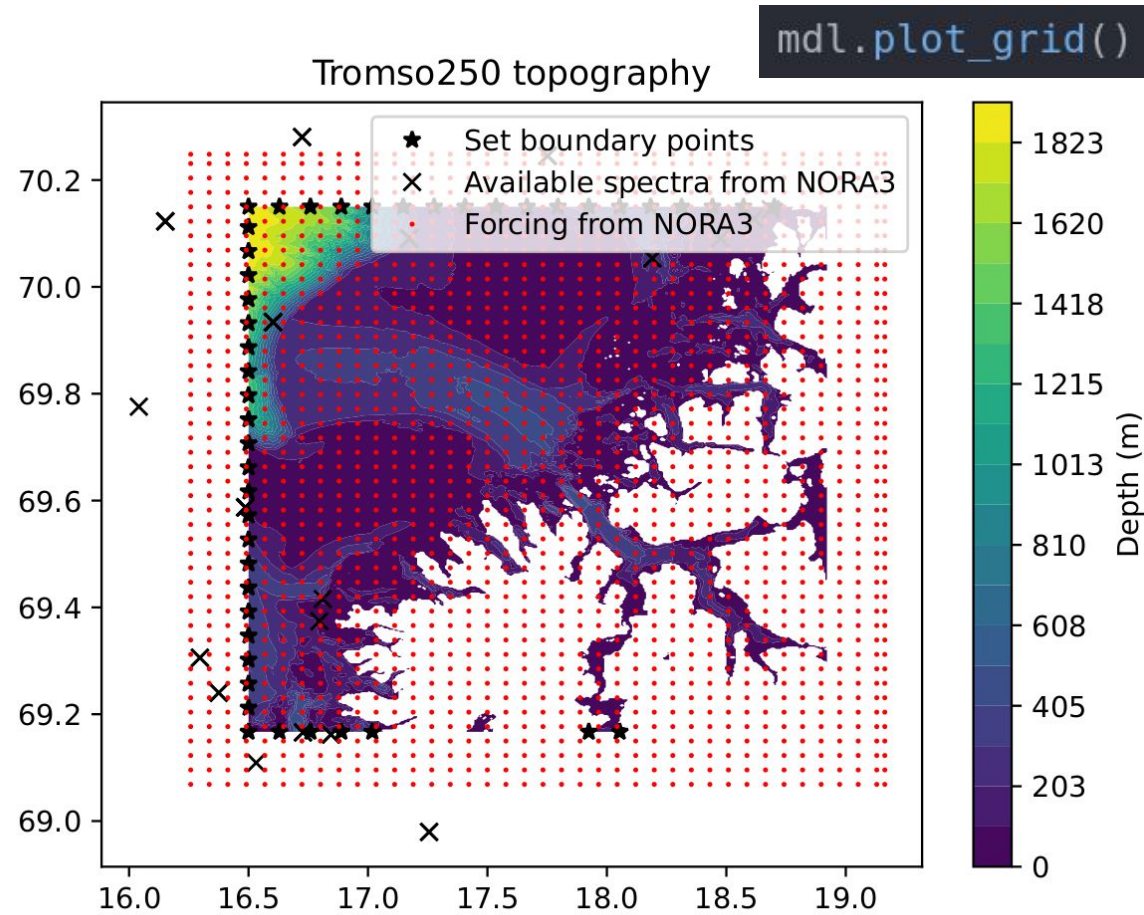


Example: DNORA/SWAN using NORA3

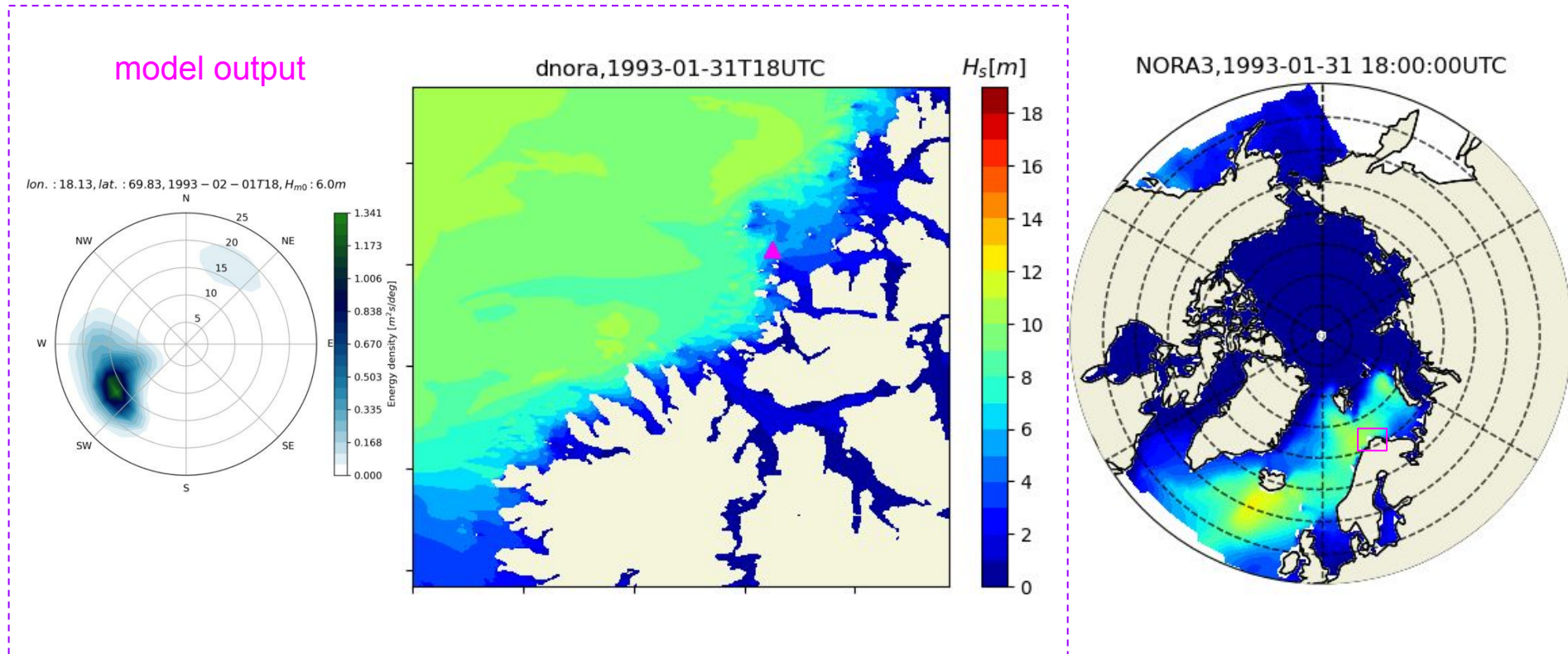
```
1 # =====
2 # IMPORT dnora
3 # =====
4 import sys
5 dnora_directory = '../'
6 sys.path.insert(0, dnora_directory)
7 from dnora import grd, mdl, inp
8 # =====
9 # DEFINE GRID OBJECT
10 # =====
11 # Set grid definitions
12 grid = grd.Grid(lon=(16.500, 18.9212), lat=(69.167, 70.150), name='Tromso250')
13
14 # Set spacing and boundary points
15 grid.set_spacing(dm=250)
16
17 # Import topography and mesh it down to the grid definitions
18
19 grid.import_topo(topo_reader=grd.read.EMODNET2018(tile='C6',
20                                     folder='/home/konstac/bathy/'))
21
22 grid.mesh_grid()
23 #
24 # # Set the boundaries
25 bnd_set = grd.boundary.EdgesAsBoundary(edges=['N', 'W', 'S'], step=20)
26 grid.set_boundary(boundary_setter=bnd_set)
27 #
28 #
```

```
29 # =====
30 # DEFINE MODEL OBJECT
31 # =====
32 model = mdl.SWAN_NORA3(grid, start_time='1993-01-31T12:00',
33                             end_time='1993-02-04T12:00')
34 # =====
35 # IMPORT BOUNDARIES AND FORCING
36 # =====
37 model.import_boundary()
38 model.import_forcing()
39 # =====
40 # PLOT GRID, FORCING AND BOUNDARIES
41 # =====
42 model.plot_grid(save_fig=True, show_fig=False)
43 # =====
44 # WRITE OUTPUT FOR SWAN RUN
45 # =====
46 model.export_grid()
47 model.export_boundary()
48 model.export_forcing()
49 model.write_input_file(input_file_writer=
50                             inp.SWAN(spec_points=[(18.13, 69.83)]))
51 # =====
52 # SWAN RUN
53 # =====
54 model.run_model()
```

Example: DNORA/SWAN using NORA3

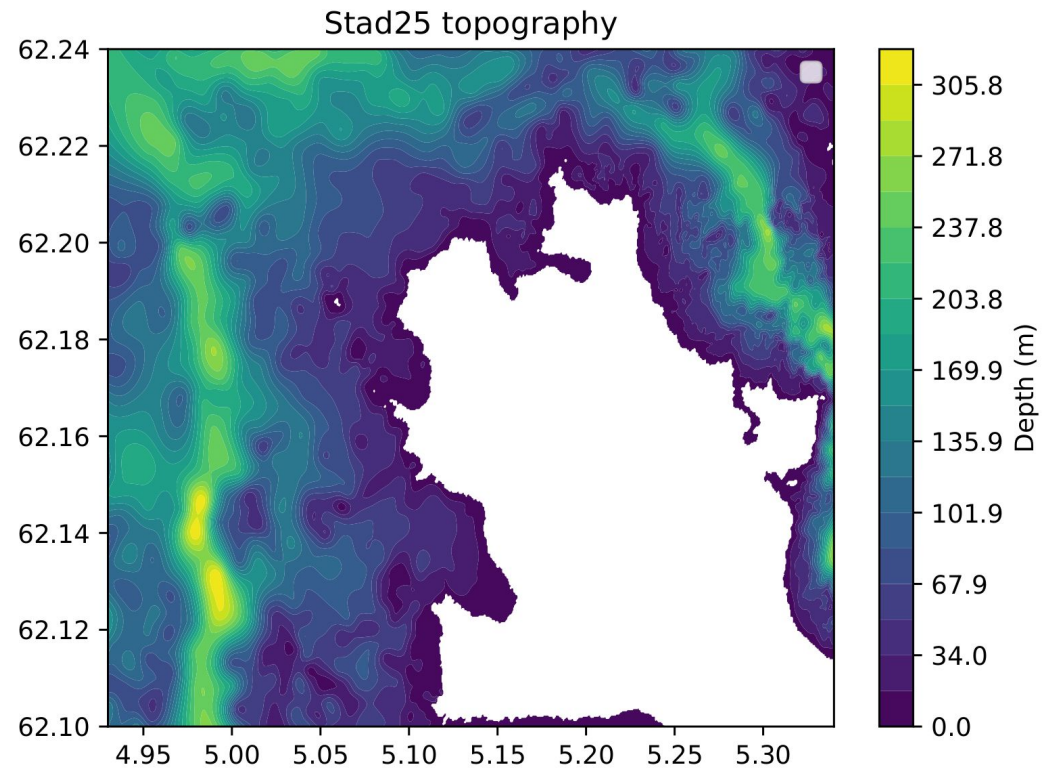


Example: DNORA/SWAN using NORA3



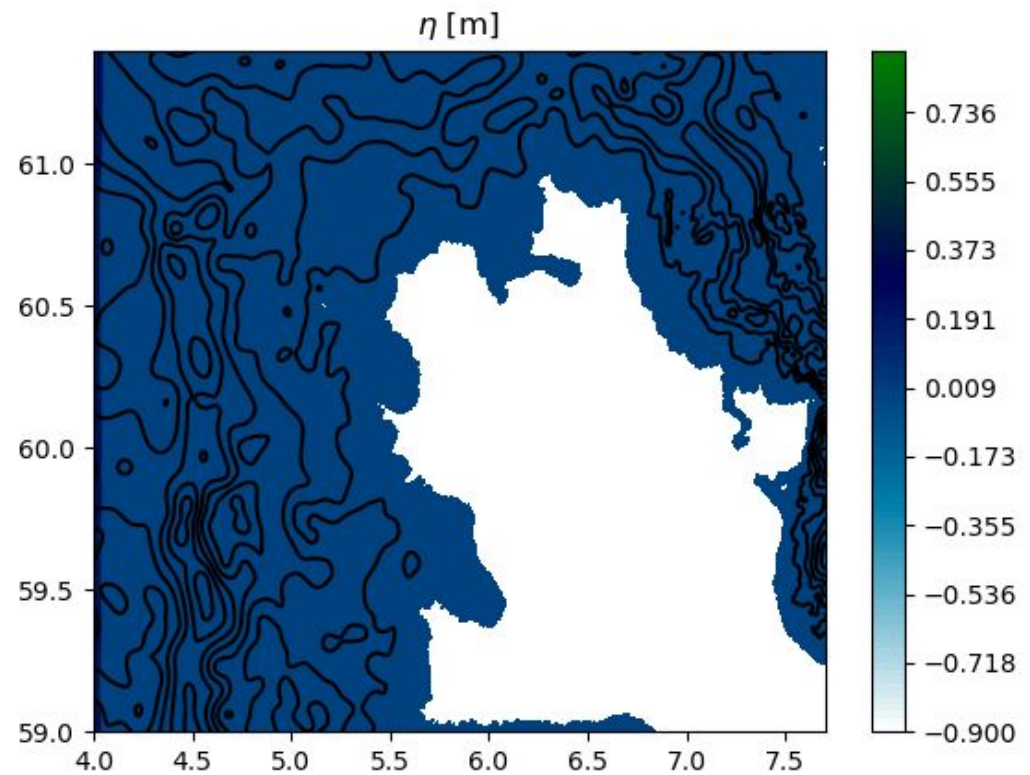
DNORA/SWASH

```
1 |
2 | # IMPORT dnora
3 | # =====
4 | import sys
5 | dnora_directory = '../'
6 | sys.path.insert(0, dnora_directory)
7 | from dnora import grd, mdl, inp
8 | # =====
9 | # DEFINE GRID OBJECT
10 | # =====
11 | # Set grid definitions
12 | grid = grd.Grid(lon=(4.93, 5.34), lat=(62.10, 62.24), name='Stad25')
13 |
14 | # Set spacing and boundary points
15 | grid.set_spacing(dm=25)
16 |
17 | # Import topography and mesh it down to the grid definitions
18 | grid.import_topo(topo_reader=grd.read.KartverketNo50m(tile='B1408',
19 |                                                       folder='/home/konstac/bathy/Kartverket'))
20 | grid.mesh_grid()
21 |
22 | # =====
23 | # DEFINE MODEL OBJECT
24 | # =====
25 | model = mdl.SWASH(grid, start_time='2018-08-25T00:00', end_time='2018-08-25T02:00')
26 |
27 | # =====
28 | # PLOT GRID, FORCING AND BOUNDARIES
29 | # =====
30 | model.plot_grid(save_fig=True, show_fig=False)
31 |
32 | # =====
33 | # WRITE OUTPUT FOR SWASH RUN
34 | # =====
35 | model.export_grid()
36 | model.write_input_file(input_file_writer=inp.SWASH(
37 |     bound_side_command='BOU SIDE W CCW CON REG 1.0 20 270 '))
38 | # =====
39 | # SWASH RUN
40 | # =====
41 | model.run_model()
42 |
```



DNORA/SWASH

```
1 |
2 | # IMPORT dnora
3 | # =====
4 | import sys
5 | dnora_directory = '../'
6 | sys.path.insert(0, dnora_directory)
7 | from dnora import grd, mdl, inp
8 | # =====
9 | # DEFINE GRID OBJECT
10 | # =====
11 | # Set grid definitions
12 | grid = grd.Grid(lon=(4.93, 5.34), lat=(62.10, 62.24), name='Stad25')
13 |
14 | # Set spacing and boundary points
15 | grid.set_spacing(dm=25)
16 |
17 | # Import topography and mesh it down to the grid definitions
18 | grid.import_topo(topo_reader=grd.read.KartverketNo50m(tile='B1408',
19 |                                                       folder='/home/konstac/bathy/Kartverket'))
20 | grid.mesh_grid()
21 |
22 | # =====
23 | # DEFINE MODEL OBJECT
24 | # =====
25 | model = mdl.SWASH(grid, start_time='2018-08-25T00:00', end_time='2018-08-25T02:00')
26 |
27 | # =====
28 | # PLOT GRID, FORCING AND BOUNDARIES
29 | # =====
30 | model.plot_grid(save_fig=True, show_fig=False)
31 |
32 | # =====
33 | # WRITE OUTPUT FOR SWASH RUN
34 | # =====
35 | model.export_grid()
36 | model.write_input_file(input_file_writer=inp.SWASH(
37 |     bound_side_command='BOU SIDE W CCW CON REG 1.0 20 270 '))
38 | # =====
39 | # SWASH RUN
40 | # =====
41 | model.run_model()
42 |
```



Thank you