CROCO Webinars #1

Python pre-processing tools for CROCO – 2.0.3

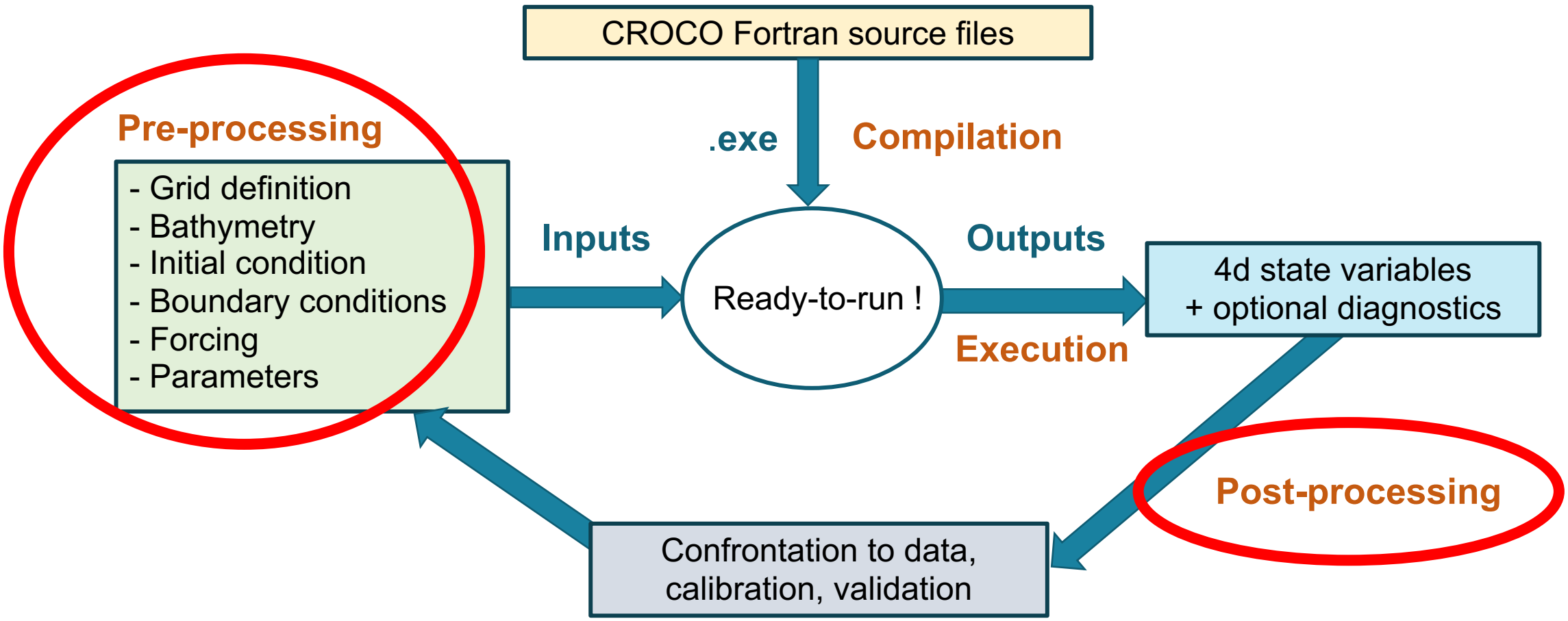
Friday, February 6th, 2026 at 1:30 PM (France time)

Presenter : Solène Le Gac (IFREMER, France)
On behalf of the CROCO pytools development team

CROCO
Coastal and Regional Ocean COmmunity model

cnrs Ifremer Inria IRD SHOM Communauté d'universités et établissements de Toulouse

Why do we need tools ?

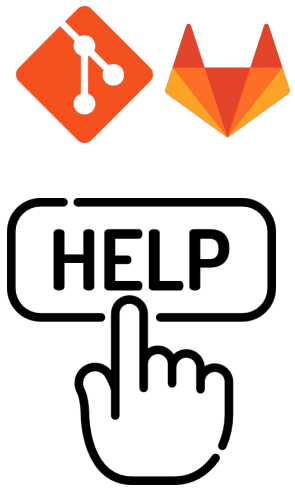


What is CROCO_PYTOOLS ?

It's a python toolbox for pre-processing, post-processing, and visualization for CROCO

It includes :

- **prepro:** preparation of input files required to run realistic CROCO simulations
- xcroco: analysis of CROCO outputs using xarray and xgcm



Code repository:

https://gitlab.inria.fr/croco-ocean/croco_pytools

Documentation:

https://croco-ocean.gitlabpages.inria.fr/croco_pytools

Forum:

https://forum.croco-ocean.org/

Installation

env.yml → python environment → f2py compilation → Ready to use

Python environment :

Pre-configured environment provided: env.yml
Includes all Python dependencies (e.g., NumPy, matplotlib, f2py)
Create and activate, example with conda:

```
conda env create -f env.yml
conda activate croco_pyenv
```

Fortran tools, interfaced through f2py : must be compiled

Some routines are written in Fortran and must be compiled
Compilation example:

```
cd prepro/Modules/tools_fort_routines
make
```

Improve on Apple chip

Reference and tutorial :

https://croco-ocean.gitlabpages.inria.fr/croco_pytools/prepro/tuto.env.html

Preprocessing pipeline

Splitting the preparation of CROCO inputs into independent steps:

Grid > Initial conditions > Boundary conditions > (Tides) > (Rivers)

CROCO_FILES READY !

What CROCO_PYTOOLS/prepro provides

- Python scripts & notebooks (`make_*.py`, `nb_*.ipynb`)
- Configurable via `.ini` files (see Examples)
- Data readers (variable mapping dictionary `readers.jsonc`)
- Download tools included (`download_*.py`)

Use separate `.ini` files per step, or a single merged configuration file for the whole pipeline

Downloading data

Scripts available for : **ERA5**, **GLOFAS**, **HYCOM** and **Mercator** products

Use -h or --help to have usage informations

Configuration file with parameters :

- to request an extent from a CROCO grid netcdf file or a specified area,
- to choose a product and the variables needed
- to request a specific period of time …

Download scripts can be long to run if you request a large amount of data

Accounts needed :

- ERA5: define your ~/.cdsapirc with key & url https://cds.climate.copernicus.eu/api
- GLOFAS: define your ~/.cdsapirc with key & url https://ewds.climate.copernicus.eu/api
- MERCATOR: provide login and password directly with command options
  or provide credentials file ~/.copernicusmarine/.copernicusmarine-credentials

Follow up copernicus API

Make grid

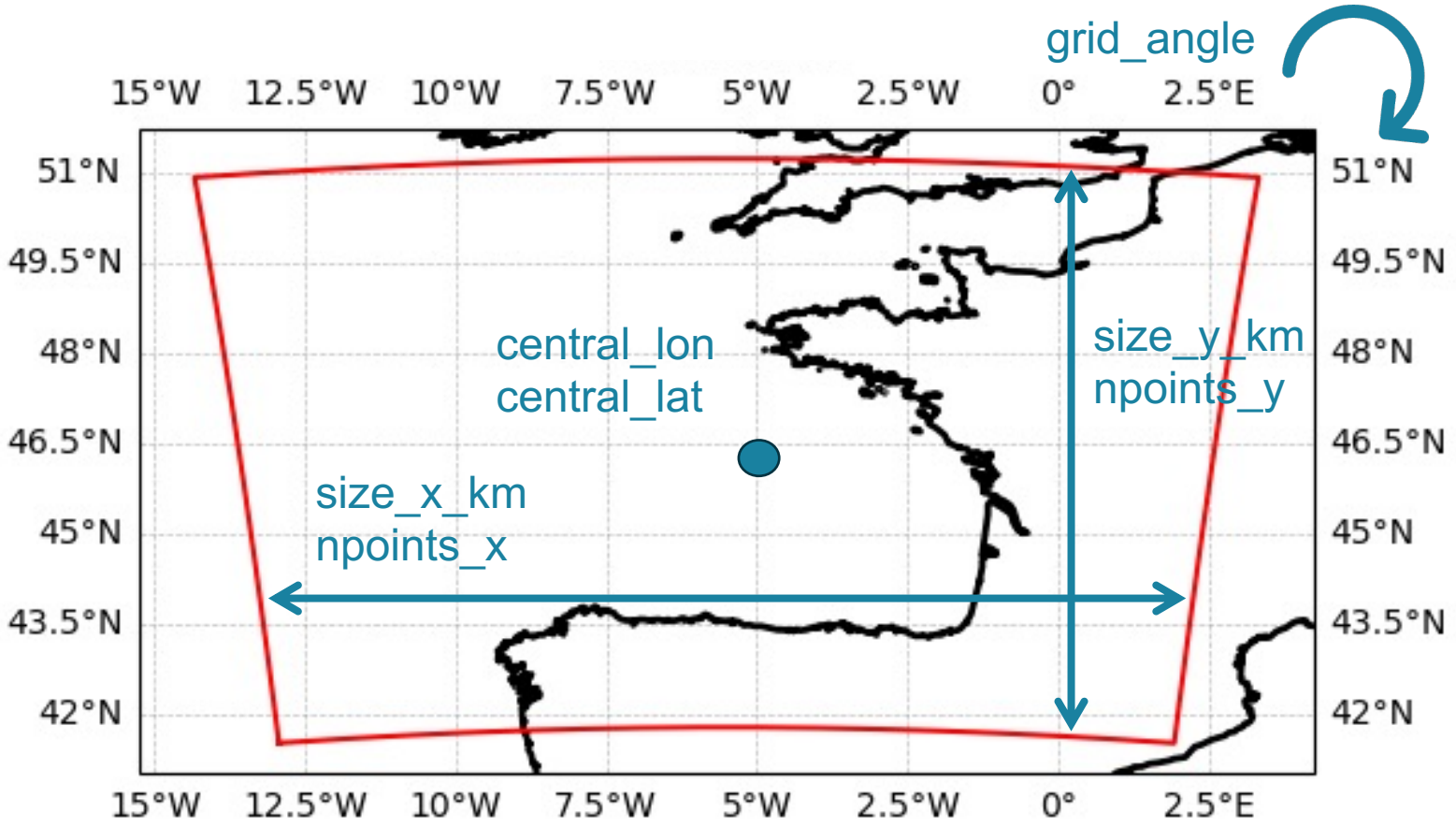
How to make a grid ? ./make_grid.py my_grid_configuration.ini

Tutorial in the documentation

To remember:

- Input data must be gridded (raster only, no vector data)
- Grid uses Oblique Mercator projection
- Several smoothing methods are available
- Isolated waterbodies can be masked

croco_grd.nc

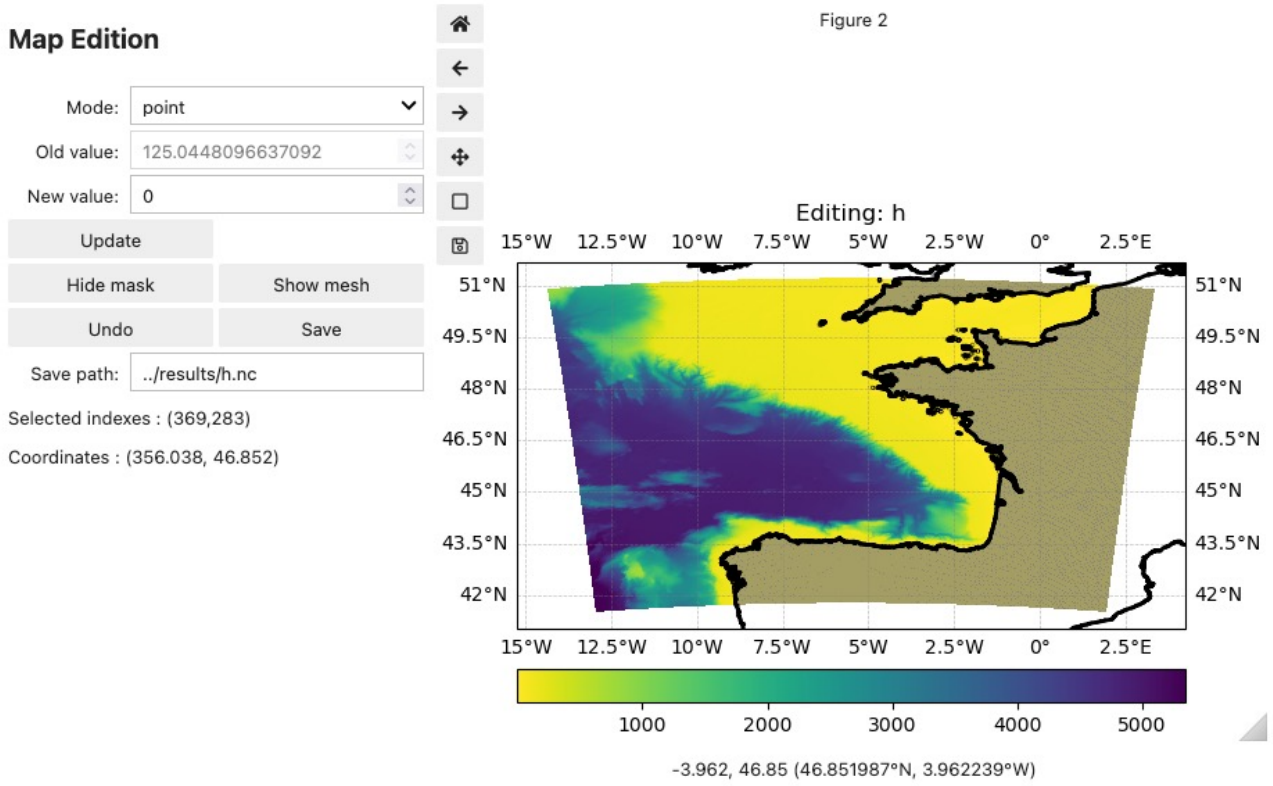


Edit the result with grid widget

How to update bathymetry « h » or mask « maskr » ? Editor demonstration

To remember :

- Point or rectangle selection
- Leave zoom mode to select a point
- Save button does NOT update croco_grd.nc directly
- After editing, bathymetry smoothing should usually be applied again



Use jupyter-forward if you are on remote computer

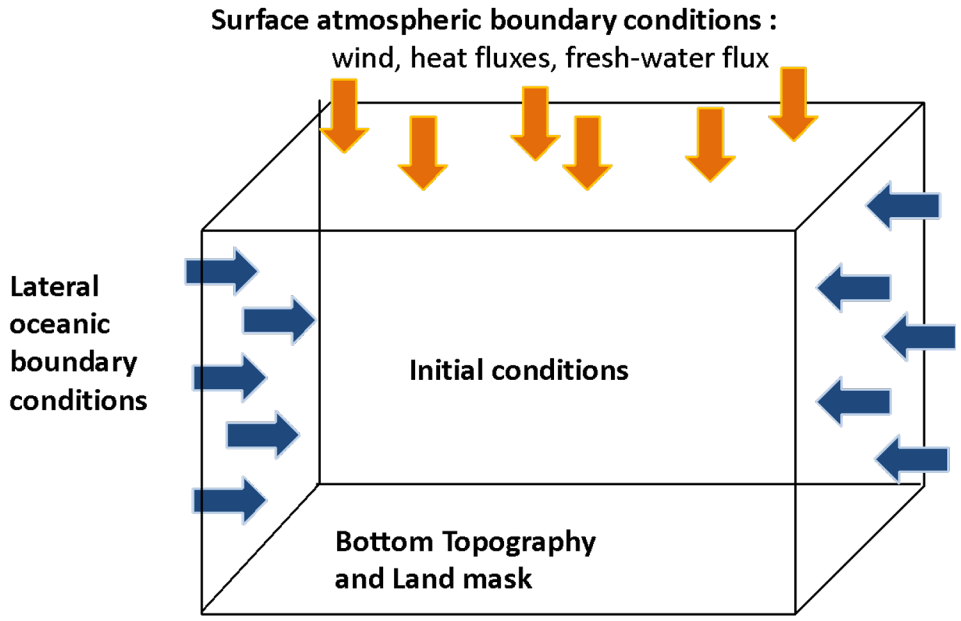
Make_ini and make_bry

How to generate the initial and boundary conditions ?
./make_ini.py my_ibc_configuration.ini
./make_bry.py my_ibc_configuration.ini

Interpolation :
Z-coordinate > S-coordinate : general case
S-coordinate > S-coordinate : zoom offline case

To remember :

- S-coordinate parameters must be coherent with CROCO parameters (N, hc, theta_s, theta_b)



croco_ini_xxx_YxxxxMxx.nc
croco_bry_xxx_YxxxxMxx.nc

NEW

Example with BGC tracers available

Make_tides

How to make tidal forcing file ?
./make_tides.py my_tides_configuration.ini

Careful with CROCO param coherency :

- Ntides
- Precompilation keys : #TIDE_MAS #USE_CALENDAR #SSH_TIDES #UV_TIDES #POT_TIDES

Check the tide atlas use : Re_Im or Amp_phase format

Check readers.jsonc file

Unless using TIDEMAS, correction at a fixed date, update to do at every restart file for long run

croco_frc_xxx.nc

All FES tide waves now available with correction

Make_rivers

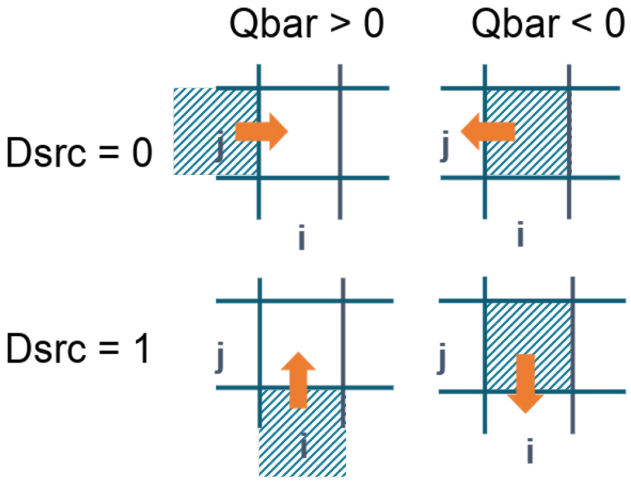
How to make runoff forcing file ?

./make_rivers.py my_rivers_configuration.ini

# input_file	Lon	Lat/Qmin
river.nc	-2.05	47.30
loire.dat	-2.05	47.30
Rhone.txt	4.82	43.37
2ddata.nc	X	10

Rivers data can have different format :

- 2D netcdf file, like GLOFAS product
- 1D netcdf file (temporal series of one river)
- 1D netcdf file (temporal series of several rivers)
- Text file (one for each river)



Make_rivers output
- croco_runoff.nc
- text file to complete croco.in psource section

Rivers widget

How to update rivers ? Editor demonstration

To remember :

- Leave zoom mode to select a river
- On save, misplaced rivers are moved
- This widget does NOT support PSOURCE_MASS option

Rivers Edition

Hide summary

Number of rivers: 5 **Number of points:** 7 **Number of splitted rivers:** 2

River	i	j	lon	lat	dsrc	qbardir
cems_glofas_ALL_0	171	17	-8.880	42.026	0	-1
cems_glofas_ALL_1	474	107	-1.454	43.620	0	-1
cems_glofas_ALL_2_split0	498	198	-0.711	45.220	1	1
cems_glofas_ALL_2_split1	497	198	-0.736	45.221	1	1
cems_glofas_ALL_3	437	305	-2.163	47.208	0	-1
cems_glofas_ALL_4_split0	517	434	0.195	49.440	0	-1
cems_glofas_ALL_4_split1	517	433	0.193	49.423	0	-1

River: None selected

Left click to select and move, right click to unselect

dsrc 0

qbardir -1

Delete selection | Add point to selecti...

Show mask | Show mesh

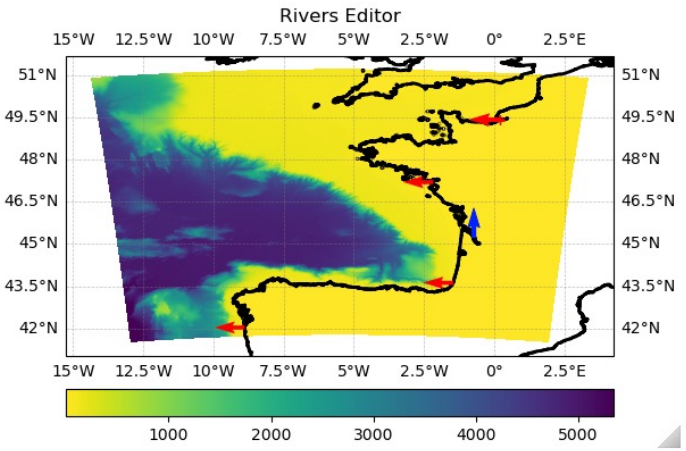
Undo

Save files

Rivers IN: ../results/river_demo/croco_runc

Rivers NC: ../results/river_demo/croco_runc

Figure 5



Use jupyter-forward if you are on remote computer

News

Last release 2.0.3 (early February 2026)

- Improve installation on Apple chip
- Update python environment to follow copernicus API change
- Add example for handling of other tracers than T,S in make_ini, make_bry
- Improve FES tide atlas handling
- Bugs fixes

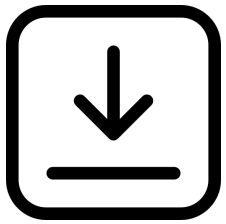
For now …

Still no widget to create grid, no multi-files handling to interpolate grid and no vector data (soundings) handling, no creation of river widget (just modification), no tracers in rivers input, only ONLINE atmospheric forcing available… … …

… Work in progress !

Feedback welcome ! https://forum.croco-ocean.org/

To conclude …



Code download

https://gitlab.inria.fr/croco-ocean/croco_pytools/-/releases

Help and details. Try Tutorials !

https://croco-ocean.gitlabpages.inria.fr/croco_pytools

Feedback welcome !

https://forum.croco-ocean.org/

Questions

Jupyter-forward example

Install jupyter-forward locally, see details in :

https://jupyter-forward.readthedocs.io/en/stable/

Example of command line :

```
jupyter-forward
        --port 8888
        --conda-env "/path_env_xxx"
        --notebook-dir "/path_nb_xxx/"
        --port-forwarding
        --shell "/bin/bash"
        --launch-command "qsub -N jlab -j oe -l select=1:ncpus=1:mem=8g -l walltime=9:00:00"
        xxx@xxx
```

Adding BGC tracers

If all tracers are in the same product, just add the tracers in the list and chck the readers.jsonc file.
If the tracers are in a different product than classical fields, make 2 downloads, see Examples/benguela_multifiles
Then in ibc.ini use prefix to indicate the right product where the variables are available

```
[IBC_Input_Files]
ibc_dir = ../data/DATA_IBC/GLORYS/
ibc_prefix = GLORYS_PHY

[Mercator_Download]
dataset = cmems_mod_glo_phy-all_my_0.25deg_P1
variables = ["uo_glor","vo_glor","thetao_glor
depths = [0,5000]
```

```
[IBC_Input_Files]
ibc_dir = ../data/DATA_IBC/GLORYS/
ibc_prefix = GLORYS_BIO

[Mercator_Download]
dataset = cmems_mod_glo_bgc_my_0.25deg_P1M-m
variables = ["chl","o2","no3","po4","si"]
depths = [0,5000]
```

```
[IBC_Options]
obc_dict = {'south':1, 'north':1,'west':1, 'east':1}
tracers = ["temp", "salt", "chl", "o2", "no3", "po4", "si"]
uv_conserv = 1
min_nb_valid_data = 4

[IBC_Input_Files]
ibc_reader = mercator_glorys_0.25deg
ibc_dir = ../data/DATA_IBC/GLORYS
ibc_prefix = GLORYS
ibc_extension = .nc
ibc_freq = 1M
ibc_multi_files = True
ibc_file_ssh = PHY
ibc_files_tracers = ["PHY", "PHY", "BIO", "BIO", "BIO", "BIO", "BIO"]
ibc_file_u = PHY
ibc_file_v = PHY
ini_filedate = Y2013M01
ini_idx = 0
```

Ini format

An INI file is a plain text file commonly used
to store configuration settings.

It is a simple and widely supported format that organizes information into sections and key-value pairs.

Example :

```
[Croco_Files]
croco_files_dir = ../results/CROCO_FILES
croco_grd_prefix = croco_grd

[Zoom_Options]
is_zoom = False
is_agrif = False
agrif_level = 0

[Grid_Position]
central_lon = 15.0
central_lat = -32.0
size_x_km = 1556
size_y_km = 1334
npoints_x = 41
npoints_y = 42
grid_angle = 0.0
```

Rivers input format

4 types :

- 2D netcdf file :
  - .nc extension
  - Qbar = variable with unit attribute in ['m3 s-1','m**3 s**-1','m3/s']
  - lon/lat : search by name and unit
- 1D netcdf file with one river :
  - .nc extension
  - Qbar = variable with unit attribute in ['m3 s-1','m**3 s**-1','m3/s']
- 1D netcdf file with several rivers :
  - .nc extension
  - Qbar = variable with unit attribute in ['m3 s-1','m**3 s**-1','m3/s']
  - lon/lat : search by name and unit
- text file : other extension than .nc

```
1950/01/01 12:00:00 730.000000
1950/01/02 12:00:00 695.000000
1950/01/03 12:00:00 675.000000
1950/01/04 12:00:00 645.000000
1950/01/05 12:00:00 610.000000
```

Readers.jsonc

```
{
    "Topo": {
      "etopo2022": {
        "lon": "lon",
        "lat": "lat",
        "topo": "z",
        "zaxis": "up"
      },
    },
…
```

```
…
    "Tides": {
      "tpxo9": {
        "lonr": "lon_z",
        "lonu": "lon_u",
        "lonv": "lon_v",
        "latr": "lat_z",
        "latu": "lat_u",
        "latv": "lat_v",
        "H_r": "hRe",
        "H_i": "hIm",
        "U_r": "uRe",
        "U_i": "uIm",
        "V_r": "vRe",
        "V_i": "vIm",
        "topor": "h",
        "topou": "hu",
        "topov": "hv",
        "invert_phase": true
      },
    },
…
```

```
…
    "IBC": {
      "mercator": {
        "depth": "depth",
        "lonr": "longitude",
        "lonu": "longitude",
        "lonv": "longitude",
        "latr": "latitude",
        "latu": "latitude",
        "latv": "latitude",
        "ssh": "zos",
        "temp": "thetao",
        "salt": "so",
        "u": "uo",
        "v": "vo",
        "time": "time",
        "time_dim": "time"
      },
    },
}
```