MANUAL FOR DATA_MED TOOLS

Contibutor : Clément Fontana

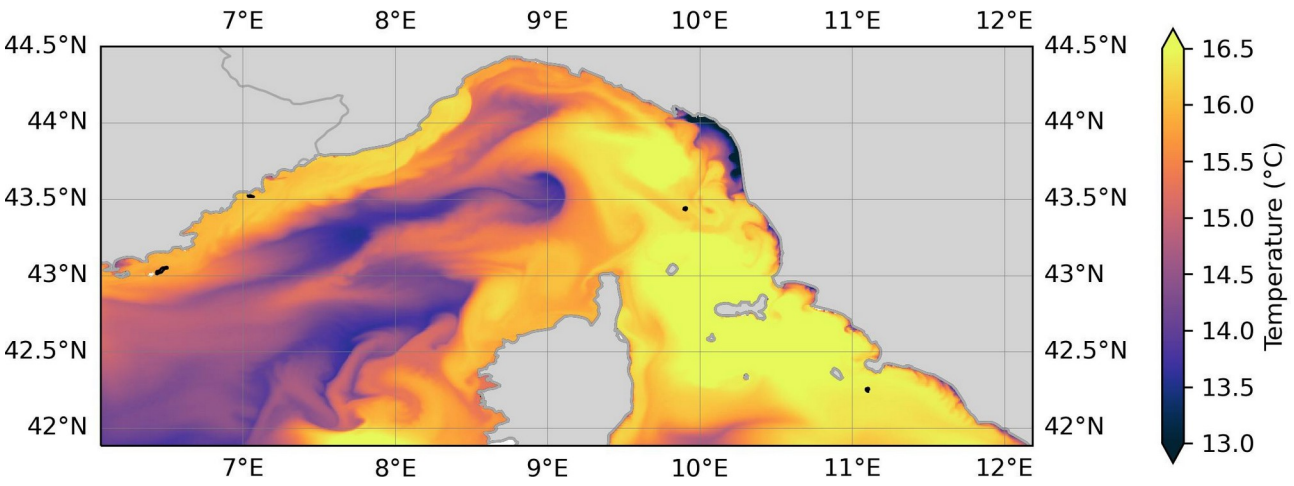


Table of Contents

contact : cfontana [a] ogs.it

1 Introduction

This document presents the architecture and instructions to process oceanographic model outputs and data in the Mediterranean Sea using the MITgcm model.

Using other models remain possible through simple nomenclature/format conversion. Contact me for any questions.

2 General arborescence

```
├── config
│   ├── config_TEMPLATE.json
│   └── variables_mit_TEMPLATE.dat
├── datasets
│   ├── BOUSSOLE
│   │   ├── data
│   │   └── read_boussole.py
│   ├── CMEMS
│   │   └── motu_downloader.json
│   ├── DVM
│   │   ├── data
│   │   ├── read_dvm.py
│   │   └── README.md
│   ├── DYFAMED
│   │   ├── data
│   │   ├── dyfamed-buoy
│   │   ├── dyfamed-mooring
│   │   ├── dyfamed-vessel
│   │   ├── read_dyfamed.py
│   │   └── README.md
│   ├── MOOSE
│   │   ├── data
│   │   ├── README.md
│   │   └── read_moose.py
│   ├── MY_NEW_DATA
│   │   ├── my_new_data.dat
│   │   └── my_new_routine.py
│   └── RIVERS
│       ├── data
│       ├── data_modif
│       ├── README.md
│       ├── read_rivers.py
│       └── readyfiles
├── gallery
│   ├── *.jpg
├── manual.pdf
├── meteo
│   ├── ECMWF
│   │   ├── get_data.py
│   │   ├── process_data.py
│   │   └── README.md
│   └── MOLOCH
│       └── process_meteo.py
├── modules
│   ├── fun_gen.py
│   ├── fun_io.py
│   ├── fun_plot_2D.py
│   ├── fun_plot_TS.py
│   └── README.md
```

```
├── plot
│   ├── make_plot_2D.ksh
│   ├── make_plot_2D.py
│   ├── make_plot_section_nongrid.py
│   └── make_plot_section.py
└── README.md
```

3 Setup

git clone git@github.com:cfontana/DATA_MED

Required Python toolbox

Core : numpy, scipy, cartopy, matplotlib, xarray

Optional : copernicusmarine (CMEMS), cdsapi (ECMWF)

Environment variables

$ set $DIR/modules to your PYTHONPATH in ~/.bashrc

$ export env variable DATA_MED_DIR as /path/to/DATA_MED

$bash

Create your own configuration file

$ cd config

$ cp config_TEMPLATE.json config_MYCONFIG.json

$ cp variables_mit_TEMPLATE.dat variables_mit_MYCONFYG.dat

(Tag MYCONFIG can be anything)

4 Parameters description

Tune your config_MYCONFIG.json file considering parameters below

```
"_____":"GENERAL PARAMETERS",

"date_ini":"2021-12-11",                          # Initial date of model data
"date_end":"2021-12-12",                          # Final date of model data
"outdir":"/path/to/OUTPUT_SMP",                   # Directory containing outputs
"diagdir":"/path/to/diags/dir",                   # Directory to save diagnostics files


"_____":"RIVER PARAMETERS",
"french_list":["Argens","Var"],                                  # French rivers list (see README)
"italian_list":["Arno","Magra","Ombrone","Serchio"],    # Italian  rivers list (see README)
"river_ini":"2021-01-01",                                        # River initial date
"river_end":"2021-12-31",                                        # River final date

"_____":"METEO PARAMETERS",
"meteo_ini":"2015-03-10",                                 # Starting date for meteo data processing
"meteo_end":"2015-03-11",                                 # Ending date for meteo data processing
"imin":"500",                                             # I index minimum cut
"imax":"600",                                             # I index maximum cut
"jmin":"350",                                             # J index minimum cut
"jmax":"450",                                             # J index maximum cut

"_____":"DYFAMED PARAMETERS",
"some_par":"some_par",              # To be defined

"_____":"BOUSSOLE PARAMETERS",
"some_par":"some_par",    # To be defined


"_____":"PLOT PARAMETERS",

"resol":"10m",                          # Cartopy coast/land resolution
"fig_proj":"ccrs.PlateCarree()",        # Cartopy projection
"fig_sx":"8",                           # Horizontal 2D figure size
"fig_sy":"8",                           # Vertical 2D figure size
"fig_fmt":"jpg",                        # Output figure format
"fig_res":"300",                        # Output figure resolution
"tight":"False",                        # Crop white space (time-consuming)
"fig_tck_size":"10",                    # Figure tick size
"fig_tcklbl_size":"10",                 # Figure tick label font size
"fig_lbl_size":"10",                    # Figure label font size
"cb_fraction_2D":"0.02",                # Python colorbar fraction parameter
"cb_pad_2D":"0.12",                     # Python colorbar pading parameter
```

```
"_____":"SECTION PLOT PARAMETERS",

"fig_secx":"8",                    # Horizontal in-depth section figure size
"fig_secy":"4",                    # Vertical in-depth section figure size
"sec_lon1":"8.5",                  # Longitude starting point for section
"sec_lat1":"43.",                  # Latitude starting point for section
"sec_lon2":"10.",                  # Longitude ending point for section

"sec_dep":"600",                   # Maximum depth for section
"sec_resH":"0.05",                 # Horizontal resolution for interpolation
"sec_resV":"1",                    # Vertical resolution for interpolation
"cb_fraction_sec":"0.025",         # Python colorbar fraction parameter for section plot
"cb_pad_sec":"0.06",               # Python colorbar padding parameter for section plot

"_____":"PROFILE PLOT PARAMETERS",

"fig_prox":"4",                    # Horizontal profile figure size
"fig_proy":"8",                    # Vertical profile figure size
"col1":"r",                        # Colors
"col2":"b",

"_____":"INTERPOLATION PARAMETERS",
"dump":"0.05",                 # Margin kept around point to fasten interpolation (5/10 grid points)
"itp_meth":"nearest",          # Interpolation method, set nearest for tests, or linear
```

The configuration is read dynamically and its variables are global inside the system once loaded

5 Variable options description

$ cp /config/variables_mit_TEMPLATE.dat /config/variables_mit_MYCONFIG.dat

Description	Name	File tag	Colormap	Log plot	Limit mode	Plot min value	Plot max value	Label	Units
Options				True/ False	set/auto				

Example	Chl	PFTC	cmc.imola	True	Set	0.05	0.25	Chlorophyll	mg.m-3

6 Plot directory routines

This directory contains routines to perform plots. Diagnostics arborescence is automatically created and name of file saved prompted.

=> *Routine make_plot_2D.py :*

Description : Plot 2D spatial maps on the domain

Arguments : *configuration_name variable_name z-level*

 Bash utilisation with *make_plot_2D.ksh*

=> *Routine make_plot_section.py :*

Description : Plot vertical section on the domain

Arguments : *configuration_name variable_name direction coordinate_value*

direction argument can be horizontal or vertical

If *direction* is horizontal, routine plots the section from '*sec_lon1*' to '*sec_lon2*' at latitudinal '*coordinate_value*'
If direction is vertical, routine plots the section from '*sec_lat1*' to '*sec_lat2*' at longitudinal '*coordinate_value*'

 Same possible bash utilisation as *make_plot_2D.py*

7 RIVERS datasets routines

This routine create river runoff input files at the MITgcm format

The data can be download from https://www.hydro.eaufrance.fr/ for french river and http://www.sir.toscana.it/consistenza-rete for italian ones.

Get csv files from both sites and save them as rivername.csv in the /datasets/RIVERS/data

$ python3 read_rivers.py NWMED
===> produces MITgcm ready to read files in /datasets/RIVERS/readyfiles

The routine raises an error if data are not continuous. In this case you must pre-process the data (*e.g.* interpolation/climatology) and save the processed file in data_modif/rivername_modif.csv

 The routine checks for existence of the file *_modif.csv before reading the original one.

8 METEO routines

ECMWF

This routine download and process CERRA data from :

https://cds.climate.copernicus.eu/cdsapp#!/dataset/reanalysis-cerra-single-levels?tab=overview

Tune "METEO PARAMETERS" in your configuration file

Go to DATA_MED/meteo/ECMWF

Run :

$ python3 get_data.py MYCONFIG

=> NetCDF files are stored in diagdir/METEO

Run :

$ python3 process_data.py MYCONFIG

=> Interpolated NetCDF files are stored in diagdir/METEO/ITP_NC

=> Interpolated binary files are stored in diagdir/METEO/ITP_BIN

Binaray files are ready to be read by MITgcm

 So far, longitude/latitude 1-D array are read from an output file :

lon,lat,levels = load_coords()

 # Mesh model lon/lat

LAT,LON = np.meshgrid(lat,lon)

For a first model initialization, you can modify this section to read lon/lat elsewhere (bathy, mask …)

9 Test data set

You can perform tests using MITgcm NetCDF outputs available here :
https://drive.google.com/drive/folders/1zeqUdXv5-6zCRw_ok9OOMxth2WrHNgRl?usp=sharing

Just set the *outdir* and *diagdir* variables in your configuration file.

10 Create your own directory

You can easily access existing function and create your own directory to process data.
For an example of use :

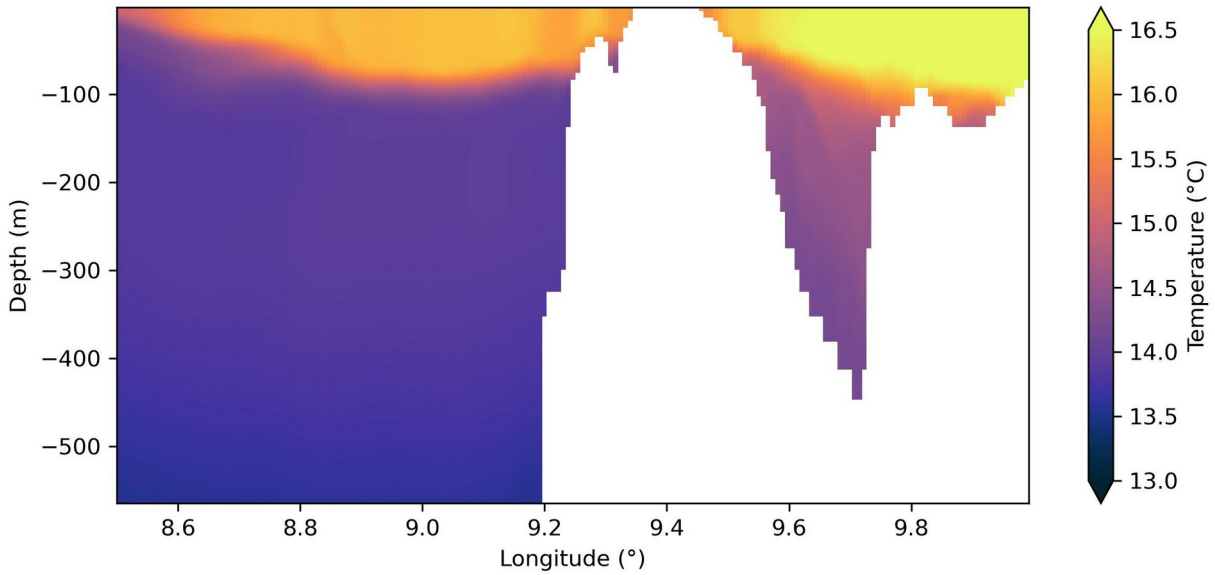
$ cd datasets/MY_NEW_DATA
$ python3 my_new_routine.py MYCONFIG

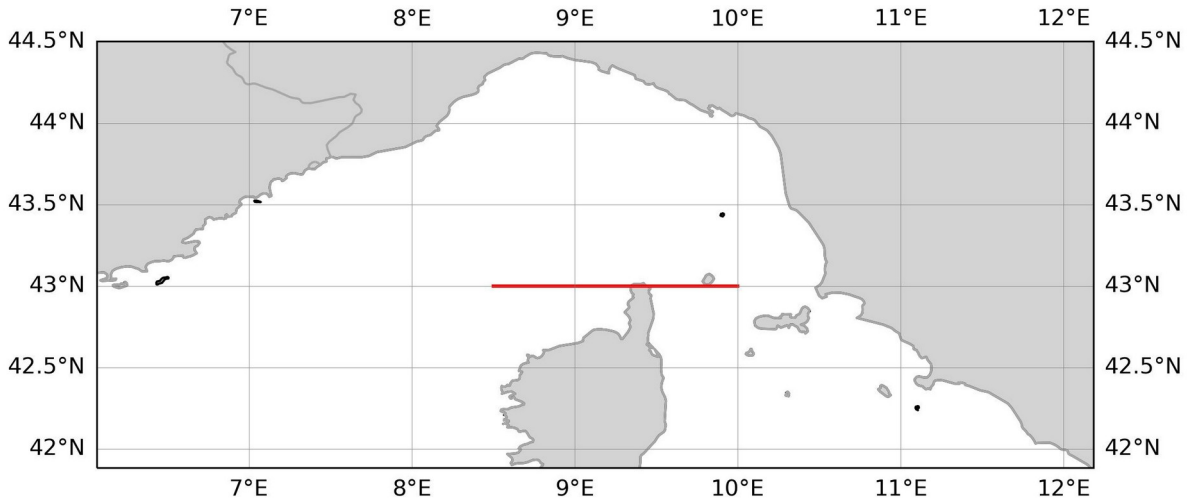
 **TIPS :**

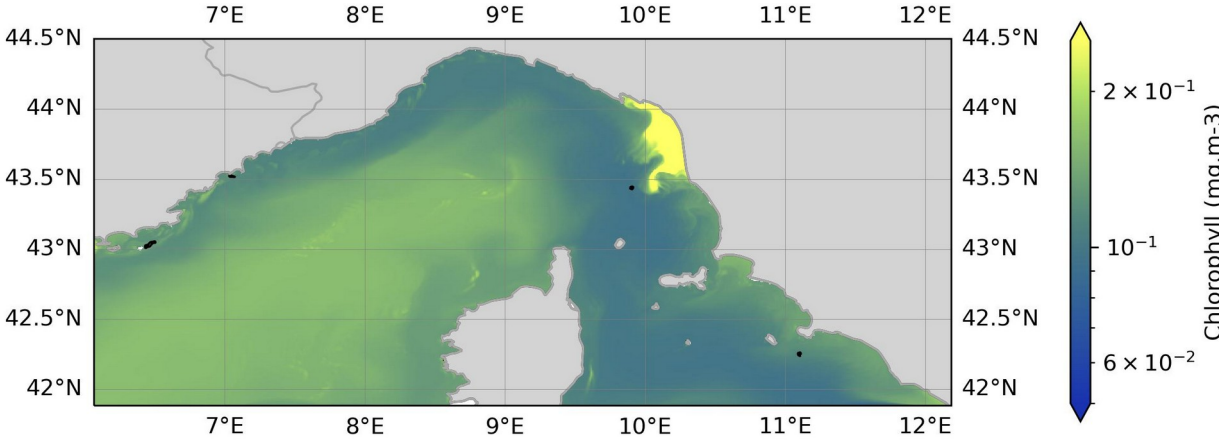
- You can add your own configuration parameter in file /config/config_CUSTOM.json and load it using load_config(CUSTOM) in your routines. We could include them in the general file later.
- You can also create your own functions in /modules and load them as others.
- Check for /module/fun_* for already existing functions (e.g. *savefig* in fun_io.py)

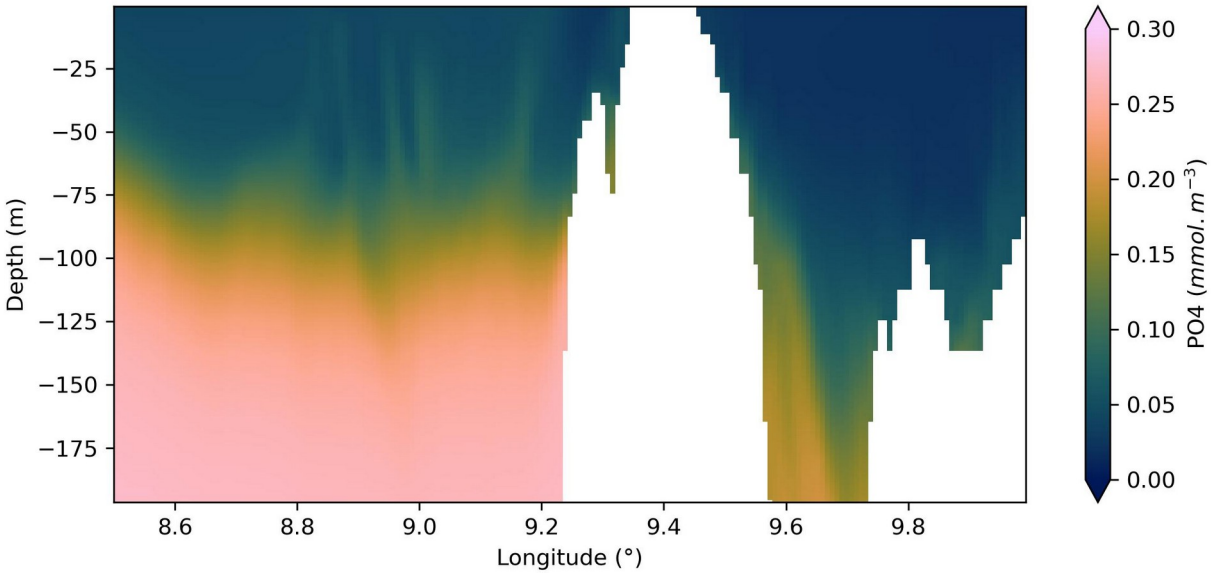
ENJOY !

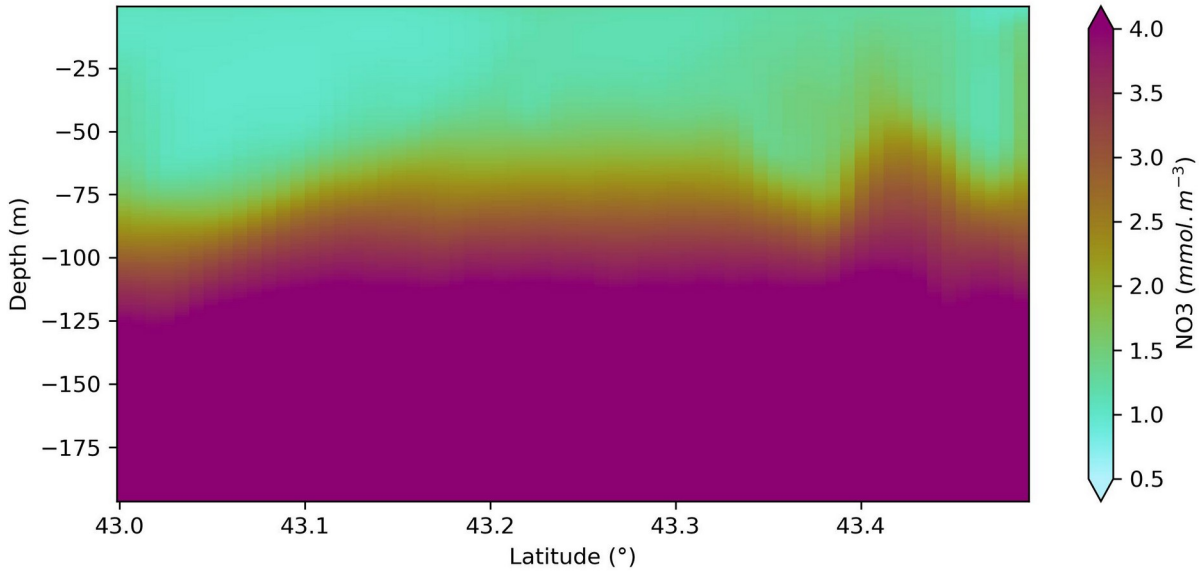
11 Gallery


Depth (m)
−100
−200
−300
−400
−500
8.6
8.8
9.0
9.2
9.4
9.6
9.8
Longitude (°)
16.5
16.0
15.5
15.0
14.5
14.0
13.5
13.0
Temperature (°C)


7°E
8°E
9°E
10°E
11°E
12°E
44.5°N
44°N
43.5°N
43°N
42.5°N
42°N


7°E
8°E
9°E
10°E
11°E
12°E
44.5°N
44°N
43.5°N
43°N
42.5°N
42°N
2×10^{-1}
10^{-1}
6×10^{-2}


Depth (m)
−25
−50
−75
−100
−125
−150
−175
8.6
8.8
9.0
9.2
9.4
9.6
9.8
Longitude (°)
0.30
0.25
0.20
0.15
0.10
0.05
0.00
PO4 ($mmol.m^{-3}$)


Depth (m)
−25
−50
−75
−100
−125
−150
−175
43.0
43.1
43.2
43.3
43.4
Latitude (°)
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
NO3 (mmol.m⁻³)