HORIZON 2020

IMMERSE

(Grant Agreement 821926)

Improving Models for Marine EnviRonment

SErvices Deliverable D7.4 *Public Report*

IMMERSE
Deliverable 7.4
Python coupling tool kit and documentation

CMCC: Laura Stefanizzi, Stefania Ciliberti, Giovanni Coppini
NOC: Thomas D. Prime, James Harle, Michela De Dominicis

1. Preamble

Task 7.3 "Prototype toolbox for seamless uptake of CMEMS products in downstream monitoring systems" is implemented by CMCC (Italy) and NOC (UK) with the scope to design new numerical tools able to facilitate the uptake of CMEMS products in downstream systems. It has direct link to WP8 which is devoted to demonstrating impact of NEMO and CMEMS evolutions on downstream studies, since supporting the creation of new tools to facilitate the setup of NEMO-based configurations and for product quality assessment.

The implementation plan is based on 3 major components:

- NEMO-Lab: it concerns the development of a Python-based suite for preparing data package to use as input for model configurations (resp. NOC)
- Research-to-Operations Interface: it concerns the preparation of interfaces with CMEMS model and observation products and with coastal models (resp. CMCC)
- IMMERSE Generic interface: it concerns the connection to WEkEO DIAS which is the state-of-the-art centralized access point to Copernicus product promoted by the EU Commission (https://www.WEkEO.eu/, further details in Section 2) for accessing/processing operational products (resp. CMCC)

This technical document is organized as follows: Section 2 presents the scopes behind the IMMERSE generic interfaces and the importance to connect new tools with WEkEO DIAS as final step; Section 3 presents the NEMO-Lab interface; Section 4 focuses on Research-to-Operations interface and its deployment as generic interface to WEkEO DIAS.

2. Towards IMMERSE Generic Interfaces

The growing of new technologies for data access and analysis requires update of research tools for easy transfer of numerical procedures in a more operational environment, able to support downstream applications for society and business. For example, the usage of CMEMS products is envisaged for coastal applications, in particular for the development of new models that use CMEMS data as lateral open boundary conditions. Or, in case of modelling assessment, the user may be interested to access insitu or satellite data from CMEMS in order to make comparison among model results and observations collected in a certain region of interest. The way these procedures are organized is crucial to optimize the access to data, to analyse the model results and to provide valuable information about the ocean state in a target region. This is the main key concept T7.3 based design and implementation of tools able to facilitate the developments of new NEMO-based model configurations thanks to the usage of CMEMS (https://marine.copernicus.eu/) products for developing new downstream services – including coastal ones – for monitoring and applications.

The WEkEO DIAS is an initiative promoted by the European Commission and implemented by EUMETSAT, ECWMF and MERCATOR OCEAN to become the EU's reference service for *harmonized data access, cloud infrastructure and expert user support*. WEKEO provides virtual environments for data processing and T7.3 started to prepare new implementations or interfaces to properly handle CMEMS

data and useful analysis for research purposes. A detailed presentation of WEkEO is provided at https://WEkEO.eu/.

T7.3 Team based integrations on WEKEO according to high level architecture as in Figure 1. WEkEO provides online catalogue access and harmonised data access API (https://www.WEkEO.eu/web/guest/hda-api) to IMMERSE: on the other side, IMMERSE implements tools for NEMO-based configurations and for processing accessed CMEMS products. IMMERSE tools have been implemented thanks to the experience in the usage of CMEMS products and in the long term supports the ingestion of new data from high resolution models and downstream services.

High level architecture is the result of the exchange between CMCC and NOC firstly, with a more extensive inclusion of users/groups that are currently working on regional and/or non-conventional NEMO configurations. A dedicated webinar and survey done in Apr 2019 highlighted the following key-points for successful implementation of new interfaces towards a more integrated one on WEkEO:

- Methods for solving high resolution ocean-to-coastal processes by improving data access and analysis
- Supporting product quality initiatives as implemented within CMEMS for assessing the new configurations
- Exploiting WEkEO capacity and filling technological gaps towards more robust and easy-to-use interfaces at research level
- Supporting transition-to-operations: concepts like interoperability and reliability are a key for next generation of tools and configurations able to support operations (and so supported by software engineering solutions).
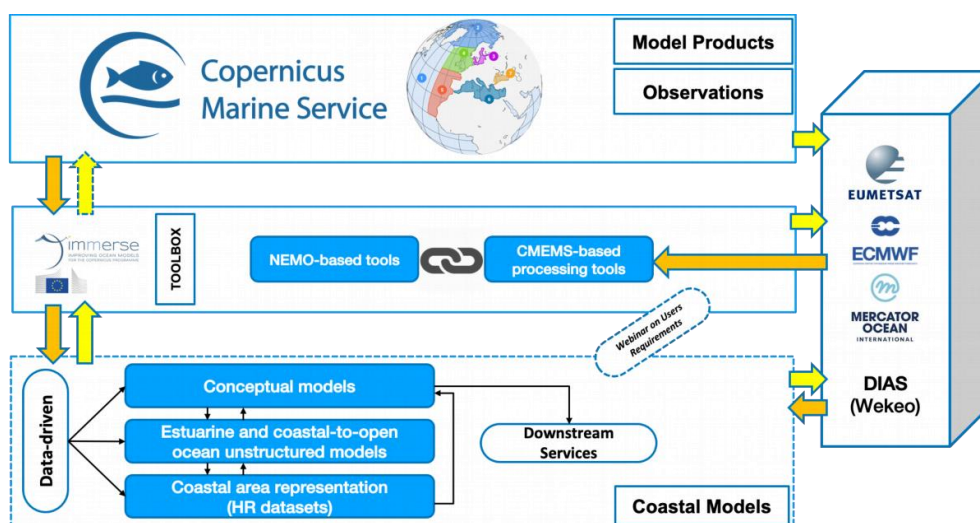

Figure 1: IMMERSE Interfaces high level architecture

3. The NEMO-Lab Interface

3.1 Introduction

The NEMO-Lab interface prepares data sources/packages to use as input for new model configurations. It builds on an existing python module - PyNEMO (https://github.com/NOC-MSM/PyNEMO) - and extends its functionality to allow users to use:

- CMEMS DL Module: to request boundary data from a CMEMS repository using a PyNEMO configuration file. The required subset of data is download and used by PyNEMO to generate model boundary forcing files.
- Tide Module: to request tide forcing is applied at the boundary. A generic module has been implemented currently for TPXO and FES tide models.
- Unit test module: to test the core and new functionality of PyNEMO

- NCML templates: to modify templates so that different inputs and outputs can be defined, e.g. new model from CMEMS or other sources such as CMIP or to match the requirements of other models e.g. FVCOM.
- Integration with DIAS (WEkEO): to use PyNEMO without any requirements to install, configure or setup PyNEMO. It would also negate the need to download data to process as this would be accessed via DIAS.

The git repo address is https://github.com/NOC-MSM/PyNEMO/tree/IMMERSE and the online documentation is located at https://pynemo.readthedocs.io/en/latest/index.html.

3.2 CMEMS DL Module

PyNEMO has a CMEMS downloading function incorporated within it, this will download a section of the CMEMS global model 'GLOBAL_ANALYSIS_FORECAST_PHY_001_024-TDS' for the defined time period in the namelist file.

To use the downloading function, the following command is used:

```
$ pynemo -d namelist.bdy
```

where the -d flag tells PyNEMO to use the CMEMS downloader and download data as specified in the namelist file. The log file that PyNEMO produces provides a log of what the downloader does. The CMEMS MOTU system is prone to disconnects and failure so there is download retry and error handling built in. Most of the options required should not need editing and are there for future use in case URL's and filenames on CMEMS change.

The options that can be configured are described in further detail in Figure 2 and commented in the next subsections. Some of the options define the behaviour of the downloader, others define locations to save files and others detail models and grid files to download. Finally, the spatial extent to download is also required.

```
!-------------------------------------------------------------------------
!  I/O
!-------------------------------------------------------------------------
   sn_src_dir = '/Users/thopri/Projects/PyNEMO/inputs/CMEMS.ncml' ! src_files/'
   sn_dst_dir = '/Users/thopri/Projects/PyNEMO/outputs'

   sn_fn       = 'NNA_R12'              ! prefix for output files
   nn_fv       = -1e20                  ! set fill value for output files
   nn_src_time_adj = 0                  ! src time adjustment
   sn_dst_metainfo = 'CMEMS example'

!-------------------------------------------------------------------------
!  CMEMS Data Source Configuration
!-------------------------------------------------------------------------
   ln_use_cmems            = .true.
   ln_download_cmems       = .true.
   sn_cmems_dir            = '/Users/thopri/Projects/PyNEMO/inputs/' ! where to download CMEMS input files (static an
   ln_download_static      = .true.
   ln_subset_static        = .true.
   nn_num_retry            = 4 ! how many times to retry CMEMS download after non critical errors?
!-------------------------------------------------------------------------
!  CMEMS MOTU Configuration (for Boundary Data)
!-------------------------------------------------------------------------
   sn_motu_server          = 'http://nrt.cmems-du.eu/motu-web/Motu'
   sn_cmems_config_template = '/Users/thopri/Projects/PyNEMO/pynemo/config/motu_config_template.ini'
   sn_cmems_config         = '/Users/thopri/Projects/PyNEMO/pynemo/config/motu_config.ini'
   sn_cmems_model          = 'GLOBAL_ANALYSIS_FORECAST_PHY_001_024-TDS'
   sn_cmems_product        = 'global-analysis-forecast-phy-001-024'
   sn_dl_prefix            = 'subset'
!-------------------------------------------------------------------------
!  CMEMS FTP Configuration (for Static Files)
!-------------------------------------------------------------------------
   sn_ftp_server           = 'nrt.cmems-du.eu'
   sn_static_dir           = '/Core/GLOBAL_ANALYSIS_FORECAST_PHY_001_024/global-analysis-forecast-phy-001-024-static
   sn_static_filenames     = 'GLO-MFC_001_024_coordinates.nc GLO-MFC_001_024_mask_bathy.nc GLO-MFC_001_024_mdt.nc'
   sn_cdo_loc              = '/opt/local/bin/cdo' ! location of cdo executable can be found by running "where cdo"
!-------------------------------------------------------------------------
!  CMEMS Extent Configuration
!-------------------------------------------------------------------------
   nn_latitude_min         = 40
   nn_latitude_max         = 66
   nn_longitude_min        = -22
   nn_longitude_max        = 16
   nn_depth_min            = 0.493
   nn_depth_max            = 5727.918000000001
```

Figure 2: DL module configuration file

3.2.1 I/O and NMCL file

The location of the NCML (NetCDF Markup Language) file is listed a string defining the source directory or "`sn_src_dir`". The output folder is also defined here as "`sn_dst_dir`". An NCML file is an XML file that is used to define an NetCDF dataset. An example of it's use is to join multiple NetCDF files such as monthly datasets so that they appear to be one large dataset (e.g. annual).

NOTE: if this directory does not exist it will need to be created and permissioned correctly for PyNEMO to run properly. The NCML file details the input files to aggregate and what the variable names are. This file can be generated using the `ncml_generator`, with variable names found using the CMEMS catalogue https://marine.copernicus.eu/.
For more information, please read the ncml generator page.

NOTE: A NCML file must be used and it also must use a regular expression. The CMEMS downloader uses this regular expression to determine what grid a given variable is part of e.g. temperature and salinity on the T grid. The example `CMEMS.ncml` file includes: an implementation of how to define temperature, SSH and U and V components of ocean currents.

Within the I/O and NCML section of the namelist config file, the string "`sn_fn`" defines the prefix for the output files. The number "`nn_fv`" defines the fill value, and the number "`nn_src_time_adj`" defines the source time adjustment. The rest of the boxes are CMEMS specific and are explained in sections 3.2.2 to 3.2.5.

3.2.2 Data Source Configuration

The first section defines the CMEMS data source configuration. The boolean "`ln_use_cmems`" when set to true will use CMEMS data as defined in the NCML file. On its own this flag just tells PyNEMO that the user wants to use CMEMS and that it is present as defined in the NCML file. If it is not present then the user needs to download the data using the CMEMS downloader, see section 3.2.3 If a user is using CMEMS data this flag should always be set to true.

3.2.3 MOTU Configuration

In the next section when set to true "`ln_download_cmems`" will download the boundary tracer data, e.g. time series of temperature and salinity. When set to false PyNEMO will skip this download. The string "`sn_cmems_dir`" defines where to save these downloaded files. PyNEMO requires grid data, this isn't possible to download using the same method as the tracer data which uses the MOTU python client. To get the grid data, an ftp request is made to download the global grids which are then subset to the relevant size. The Booleans "`ln_downlad_static`" and "`ln_subset_static`" determine this behaviour. Finally, there is an integer named "`nn_num_retry`" which defines the number of times to retry downloading the CMEMS data. The data connections are prone to failure so if a noncritical error occurs the function will automatically try to redownload. This int defines how many times it will try to do this. Typically, this static data and subsetting are only required once so these can be set to true for first download and then set to false when more time series data is required.
As mentioned previously, the time series boundary data is downloaded using MOTU, this is an efficient and robust web server that handles, extracts and transforms oceanographic data. By populating a configuration file, this can be sent to the MOTU server which will return the requested data in the requested format. The section CMEMS MOTU configuration sets this up. Most of these options should not need changing. The location of the MOTU server for CMEMS is defined here, and the location of the config template file and also the location of the config file to submit. The only options that should require changing are the model, product and prefix options. These define which CMEMS model and product to download and the prefix is a user defined string to prefix the downloads. A catalogue of the CMEMS model and products can be found at https://marine.copernicus.eu/. Currently PyNEMO has only been tested using the physical global forecast model. The downloader

should be able to download other models and products, but it has not been tested and they are known issues with other products. For example, the Northwest Atlantic model is not currently compatible due to differences in how the model variables are stored).

3.2.4 FTP Configuration for Static and Grid files

This section defines the FTP server, the remote directory and which files to download. This should not require modification unless CMEMS changes the file structure or names on the FTP server. Note: it is important that the filenames are separated by a space as this is what PyNEMO is expecting. Finally, the location of CDO executable which should be installed to enable subsetting to occur. This can be found by running:

```
$ where cdo
```

3.2.5 Extent Configuration

Finally the last box, this is where the extent to download is configured, it is up to the user to decide but it is suggested this is at least 1 degree wider than the destination or child configuration. The depth range to request is also defined here. This information can be extracted from the CMEMS catalogue. Once set for a given configuration this will not need to be edited.

3.3 Tide Module

By providing a global tidal model dataset (TPXO and FES are currently supported), PyNEMO can generate boundary conditions for a NEMO configuration, using a user supplied namelist file.

3.3.1 Namelist options

To use the namelist, it needs to be configured with the required options. These are listed below in Figure 3:

```
!------------------------------------------------------------------------
!   unstructured open boundaries tidal parameters
!------------------------------------------------------------------------
    ln_tide        = .false.             ! =T : produce bdy tidal conditions
    sn_tide_model  = 'fes'               ! Name of tidal model (fes|tpxo)
    clname(1)      = 'M2'                 ! constituent name
    clname(2)      = 'S2'
    !clname(3)      = 'N2'
    !clname(4)      = 'O1'
    !clname(5)      = 'K1'
    !clname(6)      = 'K2'
    !clname(7)      = 'L2'
    !clname(8)      = 'NU2'
    !clname(9)      = 'M4'
    !clname(10)     = 'MS4'
    !clname(11)     = 'Q1'
    !clname(12)     = 'P1'
    !clname(13)     = 'S1'
    !clname(14)     = '2N2'
    !clname(15)     = 'MU2'
    ln_trans       = .false.             !  interpolate transport rather than velocities
    ln_tide_checker = .false.            ! run tide checker on PyNEMO tide output
    sn_ref_model   = 'fes'               ! which model to check output against (FES only)
```

Figure 3 : Tide Module part of DL module configuration file

These options define the location of the tidal model datasets, note this differs depending on model as TPXO has all harmonic constants in one NetCDF file whereas FES has three separate NetCDF files (one for amplitude two for currents) for each constant. Extra harmonics can be appended to the **clname(n)** list. FES supports 34 constants and TPXO7.2 has 13 to choose from. Other versions of TPXO should work with PyNEMO but have not been yet been tested.

NOTE: FES dataset filenames must have be in the format of constituent then type - e.g.:

```
M2_Z.nc (for amplitude)

M2_U.nc (for U component of velocity)

M2_V.nc (for V component of velocity)
```

If this is not undertaken the PyNEMO will not recognise the files. TPXO data files are specified directly so these can be any name although it is best to stick with the default names as shown above. So far the tidal model datasets have been downloaded and used locally but could also be stored on a THREDDS server although this has not been tested with the global tide models.

Other options include "`ln_tide`" a Boolean that when set to true will generate tidal boundaries. "`sn_tide_model`" is a string that defines the model to use, currently only "`fes`" or "`tpxo`" are supported. "`ln_trans`" is a Boolean that when set to true will interpolate transport rather than velocities.

3.3.1 Harmonic output checker

There is an harmonic output checker that can be utilised to check the output of PyNEMO with a reference tide model. So far the only supported reference model is FES but TPXO will be added in the future. Any tidal output from PyNEMO can be checked (e.g. FES and TPXO). While using the same model used as input to check output doesn't improve accuracy, it does confirm that the output is within acceptable/expected limits of the nearest model reference point.

There are differences as PyNEMO interpolates the harmonics and the tidal checker does not, so there can be some difference in the values particularly close to coastlines.

The checker can be enabled by editing the following in the relevent bdy file:

```
ln_tide_checker = .true.    ! run tide checker on PyNEMO tide output

sn_ref_model    = 'fes'     ! which model to check output against (FES only)
```

The boolean determines if to run the checker or not, this takes place after creating the interpolated harmonics and writing them to disk. The string denotes which tide model to use as reference, so far only FES is supported. The string denoting model is not strictly needed, by default FES is used.

The checker will output information regarding the checking to the NRCT log, and also write an spreadsheet to the output folder containing any exceedance values, the closest reference model value and their locations. Amplitude and phase are checked independently, so both have latitude and longitude associated with them. It is also useful to know the amplitude of a exceeded phase to see how much impact it will have so this is also written to the spreadsheet. An example output is shown below, as can be seen the majority of the amplitudes, both the two amplitudes exceedances and the ones associated with the phase exceedances are low (~0.01), so can most likely be ignored. There are a few phase exceedances that have higher amplitudes (~0.2) which would potentially require further investigation. A common reason for such an exceedance is due to coastlines and the relevant point being further away from an FES data point.

3.3.2 Tide Checker Example Output for M2 U currents

The actual thresholds for both amplitude and phase are based on the amplitude of the output or reference, this is due to different tolerances based on the amplitude. e.g. high amplitudes should have

lower percentage differences to the FES reference, than lower ones simply due to the absolute amount of the amplitude itself, e.g. a 0.1 m difference for a 1.0 m amplitude is acceptable but not for a 0.01 m amplitude. The smaller amplitudes contribute less to the overall tide height so larger percentage differences are acceptable. The same also applies to phases, where large amplitude phases have little room for differences but at lower amplitudes this is less critical, so a higher threshold is tolerated.

The following power functions are used to determine what threshold to apply based on the reference model amplitude.

Table 1. Macro for implementing amplitude and phase thresholds

	amp_lat	amp_lon	amp	ref_amp	ref_amp_lats	ref_amp_lons	phase_lat	phase_lon	phase	phase_amp	ref_phase	ref_phase_amp	ref_phase_lats	ref_phase_lons
0	42.70304	-12.1298	0.012124	0.032651	42.6875	-12.125	63.49699	-19.4455	230.4348	0.018173	357.7578	0.021120878	63.5	-19.4375
1	42.70132	-12.0454	0.010671	0.048105	42.6875	-12.0625	64.44556	-14.3924	285.6499	0.2693642	273.9555	0.180330738	64.4375	-14.375
2							64.43611	-14.184	288.6613	0.297506	318.0232	0.112966992	64.4375	-14.1875
3							64.43132	-14.0799	289.3252	0.2513785	277.2954	0.260572553	64.4375	-14.0625
4							42.70304	-12.1298	330.5653	0.0121239	51.03762	0.032650944	42.6875	-12.125
5							62.56734	6.078155	213.9548	0.1704149	357.4835	0.288091391	62.5625	6.0625
6							55.03949	11.32306	270.1506	0.0450384	319.7564	0.038531847	55.0625	11.3125
7							56.02749	11.75804	43.86292	0.0349186	84.85762	0.017763961	56	11.75
8							57.04395	12.25711	181.7103	0.0259378	356.9073	0.026538705	57.0625	12.25
9							57.08778	12.27988	183.6528	0.0261472	356.9073	0.026538705	57.0625	12.25

3.3.3 Amplitude threshold

Important

Percentage Exceedance = 26.933 * Reference Amplitude ^ -0.396'

3.3.4 Phases threshold

Important

Phase Exceedance = 5.052 * PyNEMO Amplitude ^ -0.60

3.3.5 Future work

Create options of harmonic constants to request rather than manually specifying a list. These could be based on common requirements and/or based on the optimal harmonics to use for a specified time frame.

3.4 Unit Test Module

To test operation of the PyNEMO module, running the PyTest script in the unit tests folder will perform a range of tests on different child grids, e.g. checking the interpolation of the source data on to the child grid. To do this the following command is required :

```
$ pytest -v pynemo/pynemo_unit_test.py
```

The results of the test will show if all tests pass or the errors that result from failed tests.

There are 7 tests that cover checking the interpolation results of different child grids. The input data is generated as part of the test and is removed afterwards. The number of tests can be increased easily in the future if required.

3.5 NCML templates: use with other models and data sources

Up until now, PyNEMO was only compatible with defined inputs e.g. CMEMS and produced output that was only compatible with the NEMO model. Now PyNEMO accepts NCML files which can be configured so that different inputs can be used or the data is written out in a format suitable for a different model (e.g. FVCOM).

Examples have been included in the inputs folder on the GitHub repo, these include, local directory, THREDDS server, and CMEMS that uses the integrated downloader. Within the pynemo directory there are NCML output files that define the output NetCDF. These can be modified to create output files suitable for other models. This has not yet been widely tested but new NCML files will be added to repository as new model templates are created.

3.5 Intergration with DIAS/WEkEO (Future Plans)

PyNEMO is currently designed to be installed locally and run from a local python installation. Alternatively, it could also be installed on a HPC (e.g. ARCHER2) and run there. However, this can be quite a convoluted process, e.g. Java Runtime is required as is a python package manger such as CONDA.

To minimise this for end users, it would be beneficial to integrate PyNEMO into the operations interface. This would allow PyNEMO to be preinstalled and utilised by users without them needing to setup, install or config PyNEMO or download any CMEMS data to their system. Users could upload their NEMO configuration files (or provide openDAP links) and then run PyNEMO resulting in NEMO forcing file that could be then downloaded and used for NEMO simulations.

Alternatively, a docker image could be provided with PyNEMO already installed, the docker framework has already been used to make NEMO portable across different systems and the same process can be applied to the tools required to build the models. If Docker is installed, users could just pull the relevant container from an online repository and have PyNEMO ready to go on their system.

3.6 Example Northwest European Shelf

The GitHub repository https://github.com/NOC-MSM/PyNEMO/tree/IMMERSE contains an example input configuration, the NEMO configuration files are available on openDAP links and the boundary data can be downloaded using CMEMS downloader.
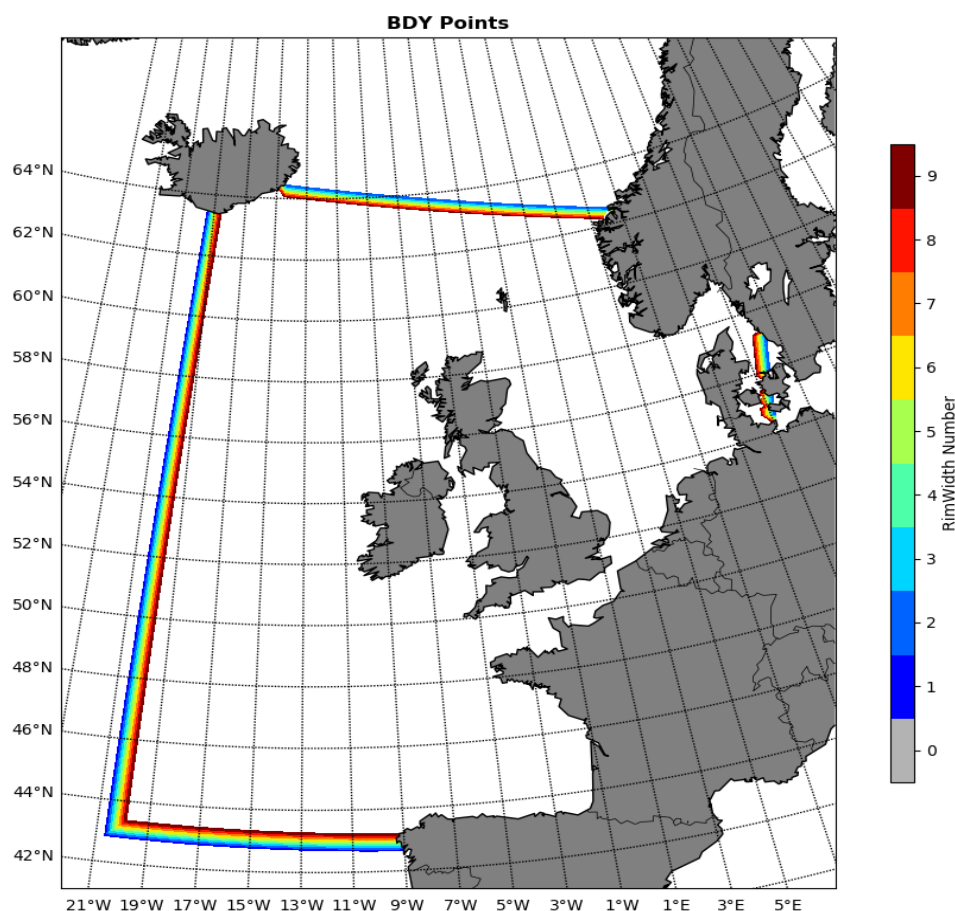

Figure 4 : Boundaries of Northwest European Shelf example.

Once PyNEMO is installed, (see online documentation
https://pynemo.readthedocs.io/en/latest/index.html) then the example will need some edits to the
namelist file that configures PyNEMO. The options that need changing are the directory paths to
match where PyNEMO is installed on the system. Here is the relevent part of the namelist file. Lines
that need changing are highlighted in yellow.

```
!----------------------------------------------------------------------------
----
!  I/O
!----------------------------------------------------------------------------
----
    sn_src_dir = '/Users/thopri/Projects/PyNEMO/inputs/CMEMS.ncml' !
src_files/'
    sn_dst_dir = '/Users/thopri/Projects/PyNEMO/outputs'

    sn_fn        = 'NNA_R12'              ! prefix for output files
    nn_fv        = -1e20                  !  set fill value for output files
    nn_src_time_adj = 0                   ! src time adjustment
    sn_dst_metainfo = 'CMEMS example'


!----------------------------------------------------------------------------
----
!  CMEMS Data Source Configuration
!----------------------------------------------------------------------------
----
    ln_use_cmems             = .true.
    ln_download_cmems        = .true.
    sn_cmems_dir             = '/Users/thopri/Projects/PyNEMO/inputs/' !
where to download CMEMS input files (static and variable)
    ln_download_static       = .true.
    ln_subset_static         = .true.
    nn_num_retry             = 4 ! how many times to retry CMEMS download
after non critical errors?
!----------------------------------------------------------------------------
----
!  CMEMS MOTU Configuration (for Boundary Data)
!----------------------------------------------------------------------------
----
    sn_motu_server           = 'http://nrt.cmems-du.eu/motu-web/Motu'
    sn_cmems_config_template =
'/Users/thopri/Projects/PyNEMO/pynemo/config/motu_config_template.ini'
    sn_cmems_config          =
'/Users/thopri/Projects/PyNEMO/pynemo/config/motu_config.ini'
    sn_cmems_model           = 'GLOBAL_ANALYSIS_FORECAST_PHY_001_024-TDS'
    sn_cmems_product         = 'global-analysis-forecast-phy-001-024'
    sn_dl_prefix             = 'subset'
!----------------------------------------------------------------------------
----
!  CMEMS FTP Configuration (for Static Files)
!----------------------------------------------------------------------------
----
    sn_ftp_server            = 'nrt.cmems-du.eu'
    sn_static_dir            =
'/Core/GLOBAL_ANALYSIS_FORECAST_PHY_001_024/global-analysis-forecast-phy-
001-024-statics'
    sn_static_filenames      = 'GLO-MFC_001_024_coordinates.nc GLO-
MFC_001_024_mask_bathy.nc GLO-MFC_001_024_mdt.nc'
    sn_cdo_loc               = '/opt/local/bin/cdo' ! location of cdo
executable can be found by running "where cdo"
!----------------------------------------------------------------------------
----
```

```
!  CMEMS Extent Configuration
!-----------------------------------------------------------------------
----
    nn_latitude_min         = 40
    nn_latitude_max         = 66
    nn_longitude_min        = -22
    nn_longitude_max        = 16
    nn_depth_min            = 0.493
    nn_depth_max            = 5727.918000000001
```

Next stage is to check the flags to download CMEMS data, as this is only required once but PyNEMO maybe run multiple times it is set to false by default. The flags are highlighted in red, in the example above there are three flags. One to download boundary data, one to download the grid (static) data and one to subset the static data, as the static data cannot be downloaded as a subset unlike the boundary data. So a user would initially set all three to true then on subsequent runs the static data flags can be set to false. If the user wanted to use same boundary data (i.e. same time period) then the download CMEMS would also be set to false.

Note: To use the CMEMS download service an account needs to be created at http://marine.copernicus.eu/services-portfolio/access-to-products/ Once created the user name and password need to be added to PyNEMO. To do this a file with the name CMEMS_cred.py in the pynemo/utils folder needs to be created with two defined strings one called user and the other called pwd to define the username and password. i.e.

```
$ touch pynemo/utils/CMEMS_cred.py
$ vim pynemo/utils/CMEMS_cred.py
user='username goes here'
pwd='password goes here'
exit and save
```

Once namelist is setup the boundary data can be downloaded using the following command (user should be in main PyNEMO directory:

```
$ pynemo -d inputs/namelist_cmems.bdy
```

The command prompt will show a spinning globe to show the downloader is running, a detailed log is saved to NRCT.log in the PYNEMO directory.

Once complete PyNEMO itself can be run with the following command:

```
$ pynemo -s inputs/namelist_cmems.bdy
```

This again will show a spinning globe with a detailed log appended to the download CMEMS log. The user may have to create the output directory in the location it is expecting - i.e., from the main PyNEMO directory.

```
$ mkdir outputs
```

These files can then be used to force a Northwest European Shelf model, (config links can be found in the namelist).

4. Research-to-Operations Interface and deployments on WEkEO (InterNEMO)

4.1 Introduction

Thanks to developed prototypes (whose first release is part of the IMMERSE MS20 – Coupler tools released for community testing), T7.3 designed the interfaces and connections with WEkEO for the usage of CMEMS products to support model developments and assessment. It is called InterNEMO – Interfaces for NEMO and allows for 3 main scopes:

1. to access and discover the CMEMS catalogue, including both model and observational data.
2. to manipulate accessed datasets to extract relevant physical information for a new NEMO-based configuration.
3. to prepare NEMO set of upstream files and to validate NEMO solution by using CMEMS observational datasets.

InterNEMO implements also technologies to connect a NEMO user to WEkEO DIAS for the interoperable accessing and processing of CMEMS data.

Considering the high dynamical evolution of technologies – including WEkEO – InterNEMO adopts *Continuous Development and Continuous Integration* approach: prototypes (source codes) are ingested according to integration requirements and transformed into a release, ready for testing and staging. Developments of interfaces and integrations for WEkEO are the baseline of IMMERSE Research-to-Operations component. Its core is represented by:

- **Access Module**: a set of drivers for the access and the discovery of the CMEMS catalogue, including modelling and observational products.
- **Process Module**: a set of numerical procedures for data manipulation (e.g., visualization, subsetting, timeseries selection).
- **Data Analysis Module**: it is an evolution of the proposed Boundary Module, coming from revision of requirements and integration of numerical procedures for facilitating the setup of new NEMO-based configurations and for evaluating the quality of numerical results.

Two kinds of interfaces have been implemented and generalized withing T7.3:

- *Standalone interface* related to user-oriented procedures. A common user, through CMEMS credentials, enters the CMEMS website and select relevant/useful information
- *Advanced interface*, representing an evolution of the standalone interface, encompassing automatic data access through WEkEO by using Harmonized Data Access API.

An overall scheme of interfaces and modules is available on Figure 5.

Such functionalities are implemented and run through Jupyter Notebook, with user-friendly GUI and exploiting advanced Python-based libraries for visualization, analysis and data retrieval.

InterNEMO is available on https://github.com/CMCC-Foundation/immerse and soon it will be cloned on https://github.com/immerse-project/.

4.2 Software Architecture Design

The whole software architecture has been designed and developed by exploiting the features offered by the Model-View-Controller (MVC) design pattern. It is one of the oldest architectural patterns for web applications. As the name suggests, the MVC Design Pattern is used to separate the logic of different layers of a program in three independent units: the Model, the View and the Controller. This is known as the principle of Separation of Concern.The three components of the MVC pattern are responsible for different things:

- the Model manages the data and defines rules and behaviors. It represents the business logic of the application. The data can be stored in the Model itself or in a database (only the Model has access to the database).

- the View presents the data to the user. A View can be any kind of output representation: a HTML page, a chart, a table, or even a simple text output. A View should never call its own methods; only a Controller should do it.

- the Controller accepts user's inputs and delegates data representation to a View and data handling to a Model.

Since Model, View and Controller are decoupled, each one of the three can be extended, modified and replaced without having to rewrite the other two. As general rule, making independent models and views makes code organization simple and easy to understand and keeps maintenance easier. To further easy Model component development, the *Intake* Python library (https://intake.readthedocs.io/en/latest/quickstart.html) has been used. It decouples data access from business logic development. Intake has been designed as a simple layer over other Python libraries to provide a consistent API that simplifies different data loading and investigation. It provides a set of data loaders (*Drivers*) with a common interface, which allow to investigate or load different kind of data with the exact same call, and turning them into well-known data structures, such as arrays and data-frames. A programmer can easy design new drivers to cover specific needs. In this project, three different drivers have been developed to manipulate Nemo-based data, SST data and Mooring data as will be introduced in the next sections. A YAML Catalogue is used for listing such data sources. It contains information about metadata and parameters and referencing which of the Drivers should load each. The View component has been developed by making an extensible use of Python *ipywidgets*. They simplify the implementation of interactive GUI for Jupyter Notebook, by improving, at the same time, the User Experience making the notebook easily usable also for not-expert users.
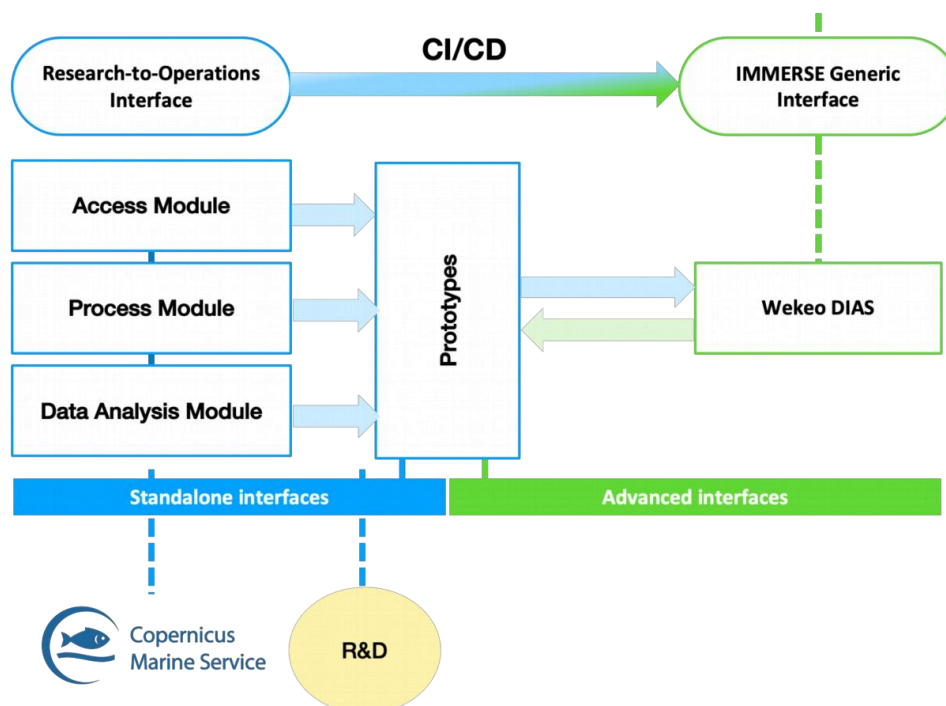
Figure 5: InterNEMO: modules, prototypes and interfaces

4.3 Access Module

The Access Module (AM) provides steps and tool for discovering the CMEMS catalogue, available at https://marine.copernicus.eu/ and on WEkEO DIAS.
InterNEMO supports 2 different implementations:
- One for the standalone version, based on MOTU Client service
- One for the WEkEO DIAS

4.3.1 Standalone version

It is based on MOTU Client service for the data retrieval and it is implemented through a Jupyter Notebook "access_module.ipynb" that recalls ad-hoc libraries for interfacing the user with CMEMS catalogue. In *Italic* an example is provided to demonstrate the main functionalities.

The main steps to access CMEMS data via Jupyter Notebook are:

1. Logging to CMEMS webportal using personal credentials

2. Catalogue Loading: at the moment, the AM is interfaced to IBI-MFC, Med-MFC and BS-MFC near real time datasets. Once the dataset is selected, the user may submit the request for loading it (*for example, we select the IBI-PHY analysis and forecast product and daily forecast dataset*)

3. Visualize Product Metadata: as part of the interactive search-and-discovery functionality, once the dataset is loaded, the user may visualize the metadata information – variables, coordinates, available time period – in order to further refine the access/selection (*considering the daily forecast dataset, available variables are 3D temperature, salinity and currents and 2D bottom temperature, mixed layer depth and sea surface height. For available variables, the system shows the coordinates and available temporal period*)

Time Coverage

```
In [7]: show_timerange(metadata)
```

#	timeCoverage
0	2020-08-01T12:00:00.000Z
1	2018-07-14T12:00:00.000Z

4. Download File: the user may select her/his own custom dataset – subsetting in space and time – and proceed with the download of the dataset (*in the example, we select just surface temperature fields up to 10m over 1 week period – from 01/07/2020 to 07/07/2020, and the file is downloaded once the user selects it. The output file is saved in a NetCDF file called "out_ibi.nc"*)

4.3.2 WEkEO-based version

It is based on integrated API to use WEkEO service for the data retrieval and it is implemented through a Jupyter Notebook "access_module_WEkEO.ipynb" that recalls ad-hoc libraries for interfacing the user with CMEMS catalogue. In *Italic* an example is provided to demonstrate the main functionalities. The main steps to access CMEMS data through WEkEO via Jupyter Notebook are:

1. Importing Modules: the core for the interface is AMController, which implements Harmonized Data Access API via notebook in an automatic way (i.e., bypassing the user's need to download, integrate and execute the provided API via WEkEO)

2. Load and Visualize Product Catalogue: it performs the same steps as in the standalone version, but accommodating WEkEO requirements. The AMController returns the list of services and products as configured to the user, which may select an load the preferred ones (*in this case,*

15

we selected the Mediterranean Sea Analysis and Forecasting Physical product and corresponding monthly mean dataset for temperature. The code returns also the short description as available in the CMEMS catalogue)

2. Load and Visualize Product Catalog

At now, a partial catalog is visualized.

```
In [2]: c = AMController()
        c.display_catalog()
```

Services: MEDSEA_ANALYSIS_FORECAST

Products: EO:MO:DAT:MEDSEA_ANALYSIS_FORECAST_PHY_006_013:med00-cmcc-tem-an-fc-m

Load Metadata

Mediterranean Sea Physics Analysis and Forecast

'''Short Description:''' The physical component of the Mediterranean Forecasting System (Med-Currents) is a coupled hydrodynamic-wave model implemented over the whole Mediterranean Basin. The model horizontal grid resolution is 1/24° (ca. 4 km) and has 141 unevenly spaced vertical levels. The hydrodynamics are supplied by the Nucleus for European Modelling of the Ocean (NEMO v3.6) while the wave component is provided by Wave Watch-III; the model solutions are corrected by a variational data assimilation scheme (3DVAR) of temperature and salinity vertical profiles and along track satellite Sea Level Anomaly observations. '''Product Citation:''' Please refer to our Technical FAQ for citing products.http://marine.copernicus.eu/faq/cite-cmems-products-cmems-credit/?idpage=169

3. Logging using WEkEO credentials: once the user has selected the preferred dataset, before proceeding he/she needs to insert the WEkEO credentials (they can be created though https://www.WEkEO.eu/web/guest/user-registration) (*in the example, we used our credentials: you should include yours once created on WEkEO, otherwise the system does not allow you to download any dataset*)

3. Logging in using Wekeo credential

Log in by using your Wekeo User Credential. If you don't have them please visit here

```
In [3]: c.display_login()
```

Username: sciliberti   Password: ••••••••••••

Log in

```
Copernicus_General_License Terms and Conditions already accepted
```

4. Discovering and download the product: the controller implements additional functions for visualizing dataset metadata. In particular, the used API allows for visualizing specific information for provided dataset, such as bounding box, temporal data range, list of variables and list of depths (for 3D variables) (*in this case, it is just visualized the bounding box of the Mediterranean Sea product; then, the user may set a specific bounding box to perform the subsetting in space and time for the selected variables*)

4. Discovery and Download File

Set the parameters for downloding.

You can use the product information visualized in this section to set the parameters needed for downloding.

```
In [4]: c.display_mtd()
```

Bounding Box	Date Range	Variables	Depths

#	lat_min	lat_max	lon_min	lon_max
0	-17.2916660308837890625	45.979167938232421875	36.291667938232421875	30.1875

The download is launched once the user specifies the preferred bounding box, temporal coverage, layer and variable(s):

To verify the correctness of the download process, the run process returns a list of messages, including estimation of the file size, processing time and status

4.4 Process Module

The Process Module (PM) implements some basic functionalities for map visualization and data analysis (at the moment, min, max and mean functions). It is interfaced to AM - both versions as introduced in the previous section - through the output NetCDF file the user built for her/his own purpose. It is implemented in the Jupyter Notebook "process_module.ipyn". In *Italic* an example is described to help the user in the first execution.

Steps to visualize and manipulate CMEMS data are the following:

1. Importing Modules: the core for the interface is PMController, which implements functions for easily interface the notebook to standalone or WEkEO-based data access, including R&D products, running in an automatic way:

2. Select file: the user may select the accessed file previously downloaded though the AM (*in the case of the example, it is so.nc, which is the salinity field for the Black Sea*)

Select File

Select the file containing the variables of interest, among those downloaded via the Access Module.

```
In [2]: c = PMController()
        c.display_files()
```

File: so.nc

3. Plotting and Manipulating the available dataset: the user may load the NetCDF file in a predefined way or she/he may ask for the plot of minimum or maximum or mean. Visualization is integrated using **hvplot** function, that allows interaction with the 2D map to get the spatial value of the selected variable and the possibility to scroll over the available temporal period (*the Black Sea domain is shown, with average salinity over the selected period. The balloon shows a selected point and corresponding salinity value*). This function can be used also for setting initial conditions and boundary conditions for a NEMO-based configuration.

Plot 2D Map

In this section you can plot the 2D map of the variables of interest. In particular, 4 different types of plots are available:

- simple : plot the 2D map of the variable of interest for each timestep and for each depth
- mean/min/max : plot the mean/min/max of the variable in the period of interest
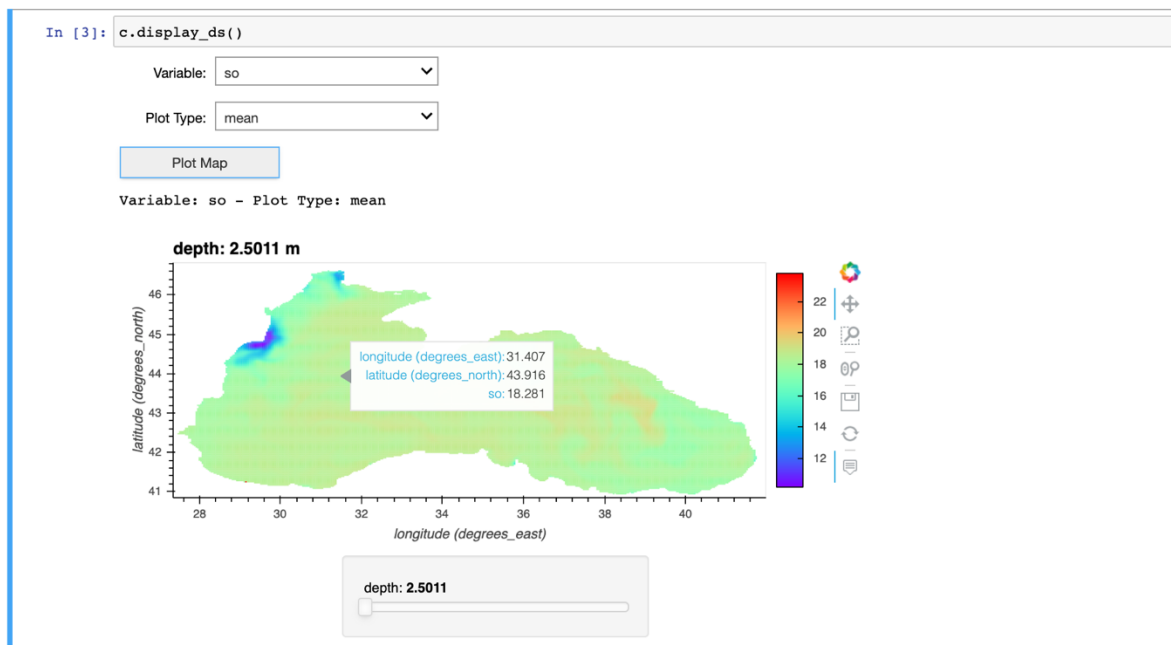

Recently, the possibility to plot the timeseries of the selected variable over a specific period and at a specific depth has been included. Also the timeseries is managed through hvplot (*in the example, we are plotting the salinity for the Black Sea at about 2.5 m – first level – over the period 03/03/2021 to 05/03/2021*).

Plot Timeseries

In this section you can plot the timeseries of the variables of interest.
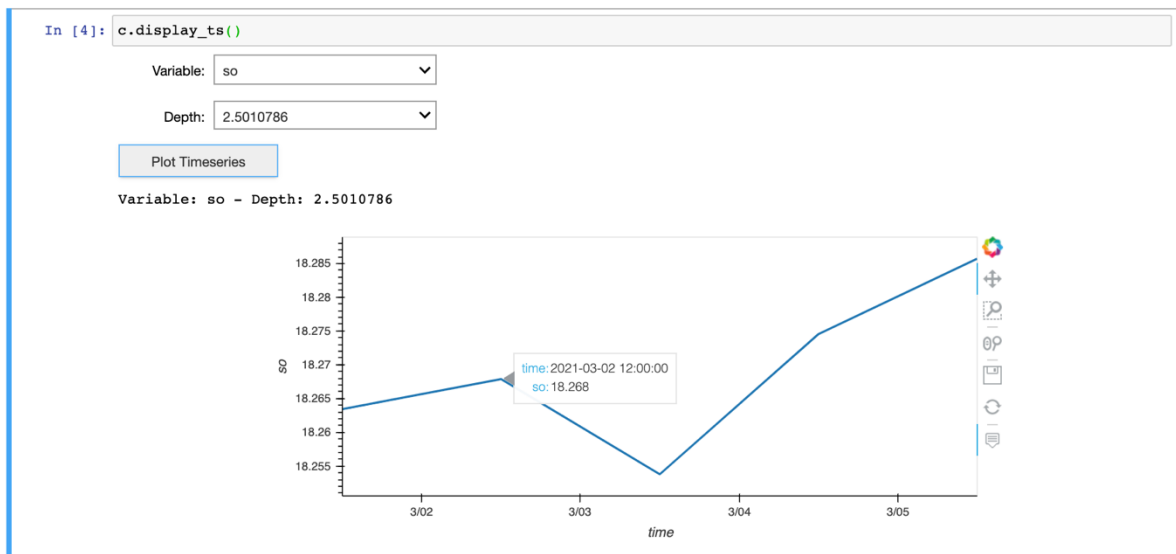
4. Save variable in a NetCDF file: the user may save the selected/manipulated variable in a new NetCDF file to further process/use it

4.5 Data Analysis Module

With respect to release v0 – Aug 2020, InterNEMO is now able to support additional functionalities for the analysis of data a) from CMEMS or b) from R&D based on NEMO for setting a new configuration as supported by PyNEMO (as described in Section 3) and for the assessment of the numerical results. The latter functionality completes the offer of the prototype in terms of usage of accessed data through the AM. To demonstrate the capacity in using accessed data, a validation exercise has been implemented and made available to user. Dedicated notebooks have been implemented for performing the evaluation of model results against SST satellite observations and INS data from moorings, both available in the CMEMS catalogue. To access local available data, once downloaded, Intake Python library is used: the user, via configuration file, may easily provide general description of relevant datasets to analyse (*in the example reported in the following, the user specifies the object "sst", represented by L3S GHRSST nighttime subskin SST for the Black Sea, whose data are available in the CATALOG_DIR, and the object "model_operational" which is the NEMO native data for the Black Sea Physics Analysis and Forecasting System as delivered through CMEMS*):

```
sst:
  description: L3S GHRSST nighttime subskin SST
  driver: drivers.SSTSource
  metadata:
    coords:
      latitude: lat
      longitude: lon
      time: time
    variables:
      temperature: sea_surface_temperature
      temperature_qc: quality_level
  args:
    urlpath: '{{ CATALOG_DIR }}files/sat/{date:%Y%m%d}000000-GOS-L3S_GHRSST-SSTsubskin-night_SST_HR_NRT-BLK-v02.0-fv01.0.nc'

model_operational:
  driver: drivers.NemoSource
  metadata:
    coords:
      latitude: nav_lat
      longitude: nav_lon
      depth: deptht
      time: time_counter
    variables:
      temperature: votemper
      salinity: vosaline
  args:
    urlpath: '{{ CATALOG_DIR }}files/products/t_{date:%Y%m%d}_dm_CMCC_BSFS1b_BLKSEA_b*_an12-v01.nc'
```

Figure 6 : Example of catalog.yaml used by the Data Analysis Module

4.5.1 Evaluating SST

A dedicated Jupyter Notebook has been implemented with the scope to perform a comparison between NEMO-based native data results and SST from satellite observations (LS3). This template can be easily adapted for performing other kind of comparisons including biogeochemistry and waves products. Provided example is for the Black Sea physical model product.

Steps are:

1. Configuration of the catalog.yaml file as previously introduced
2. Import modules, load data from the product catalogue and compute metrics: as for AM and PM, also in this case a SSTController has been implemented to collect libraries and procedure as needed for the validation exercise. In the step 2 – Load and Visualize Product Catalogue – the user specifies the source from observations "sst" and from model "model_operational" as provided in the configuration file. Implemented metrics are BIAS and RMSE (CLASS4 metrics).
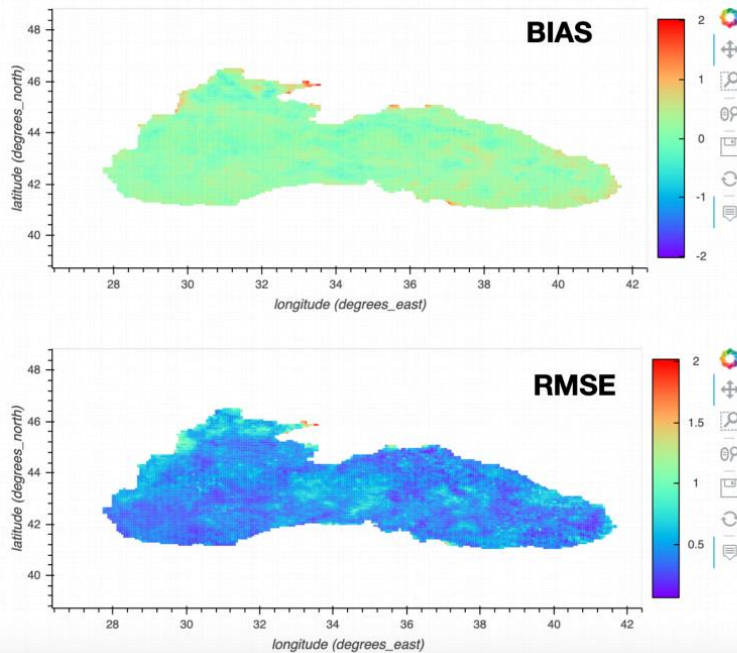

3. Visualizing the results: using hvplot library, the notebook allows for visualizing computed metrics as 2D maps averaged over the reference period and also as timeseries over the reference period.
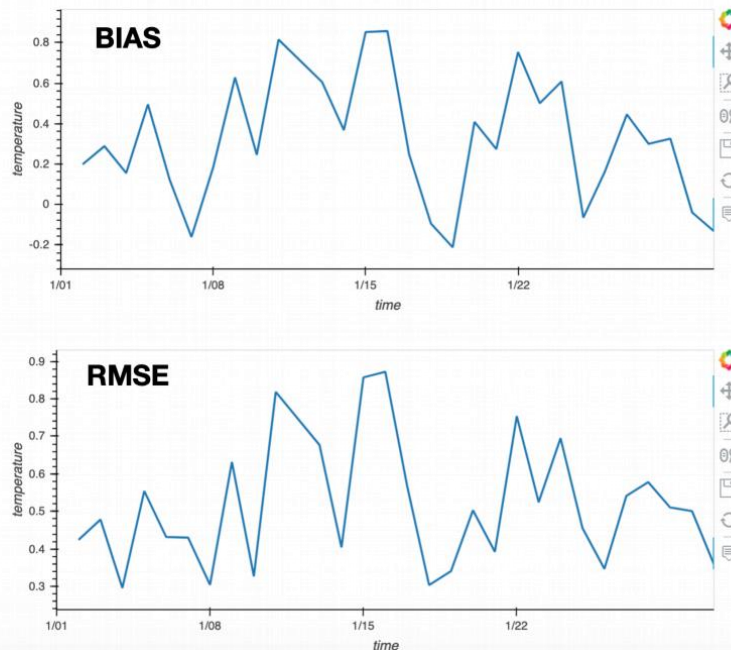
3. Display 2D map

```
In [3]: c.display_map()
```


3. Display Timeseries

```
In [4]: c.display_ts()
```


4.5.2 Evaluating T and S though INS mooring data

A dedicated Jupyter Notebook has been implemented with the scope to perform a comparison between NEMO-based native data results and temperature and salinity from INS observations. This template can be easily adapted for performing other kind of comparisons including biogeochemistry and waves products. Provided example is for the Black Sea physical model product.

Steps are:

1. Configuration of the catalog.yaml file as previously introduced
2. Import modules, load data from the product catalogue and compute metrics: as for AM and PM, also in this case a MOController has been implemented to collect libraries and procedure

21

as needed for the validation exercise, including the reading of the mooring data. In the step 2 – Load and Visualize Product Catalogue – the user specifies the source from observations "EUXRoO1" and from model "model_operational" as provided in the configuration file. The user must specify also start and end date of the validation exercise. Implemented metric is CLASS2 based, so overlapping of model and observation timeseries. The example refers to a station in the Black Sea Western basin and comparison is done against analysis field from the Black Sea Analysis and Forecasting system.

1. Import Modules

As first step you need to import all modules used in this notebook, implemented in a separate file.

```
In [1]: from MOController import *
```

2. Load Catalog

```
In [2]: c = MOController()
        c.display_catalog()
```

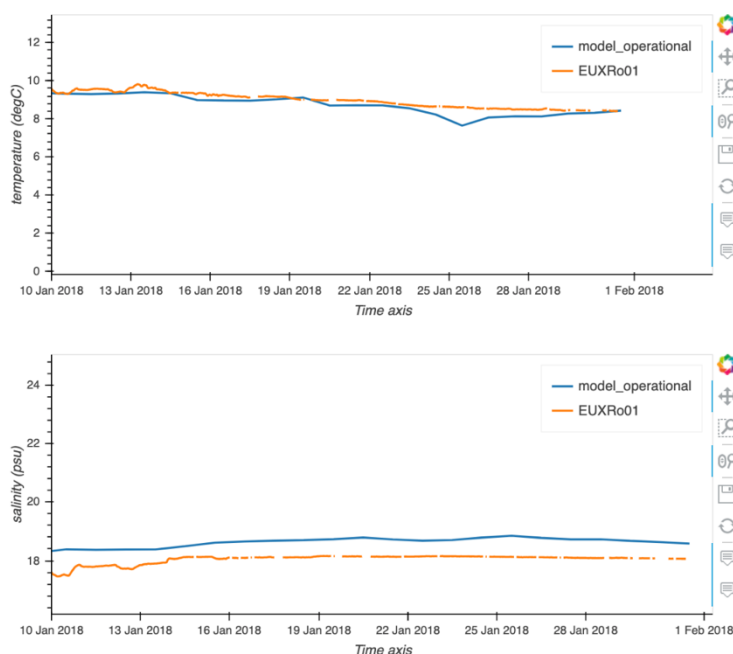|	
Model:	model_operational
Station:	EUXRo01
Start Date	01/01/2018
End Date	01/02/2018

Select

3. Display timeseries: a visualization of the timeseries for temperature and salinity from the Black Sea model against observed data at the nearest observation location is provided using hvplot.

3. Display Timeseries

```
In [5]: c.display_ts()
```


5. Conclusions

In this report, we presented implementations of new Python coupling developed within T7.3 of IMMERSE project to support users in the research community in setting and assessing new NEMO-

based configurations using CMEMS product catalogue (model and observation products). Toolkits have been developed progressively from a research-oriented approach (pyNEMO) to a research-to-operations approach (InterNEMO) for the definition of generic interfaces with CMEMS products. pyNEMO is a Python-based tool that helps to build a NEMO regional configuration, allowing for a) generation of open boundary conditions, b) usage of local or remote data sources (e.g., CMEMS via MOTU or standalone models like FES and TPXO). A unit testing approach is implemented in order to test module functionality, helping in the development and integration phase. Starting from the consolidated experience of pyNEMO, a generalized approach to access and use CMEMS data for NEMO-based model developments has been developed with InterNEMO. It is a Python-based suite that consists of 3 modules: a) Access Module for accessing and discovering CMEMS catalogue via CMEMS services (e.g., MOTU) and via WEkEO DIAS; b) Process Module, to manipulate accessed dataset including coastal ocean data, c) Data Analysis Module that performs preparation of upstream dataset for NEMO-based configuration and perform validation exercises. InterNEMO generates automatically the API request, using the Harmonized Data Access API function provided by WEkEO DIAS and JSON parser. This is part of the new functionality implemented within the Access Module to have a) centralized access point for CMEMS data, b) integrated Python-based codes and libraries to use for generalized interfaces, c) to exploit easy deployments of numerical procedures thanks to Model-View-Controller approach and d) easy transfer from R&D to operational environment thanks to flexible interfaces able to perform processing and data analysis. For these reasons, future works will be dedicated to strength pyNEMO capacity within the WEkEO DIAS framework to demonstrate impact of NEMO and CMEMS evolutions on downstream studies, which will be one of the major objective within WP8.

6. References

- Copernicus Marine Environment and Monitoring Service: https://marine.copernicus.eu/
- WEkEO DIAS: https://www.WEkEO.eu/
- Harmonized Data Access API: https://www.WEkEO.eu/web/guest/hda-api
- InterNEMO: https://github.com/CMCC-Foundation/immerse
- PyNEMO: https://github.com/NOC-MSM/PyNEMO/tree/IMMERSE
- IMMERSE H2020 Project: https://immerse-ocean.eu/