pyNEMO Documentation

Release 0.1

James Harle, Srikanth Nagella, Shirley Crompton

May 27, 2021

Contents

Contents:

CHAPTER 1

Introduction

The NRCT is a tool to set up the lateral boundary conditions for a regional NEMO model configuration. The tool is written in Python, largely within the Anaconda environment to aid wider distribution and to facilitate development. In their current form these tools are by no means generic and polished, but it is hoped will form a foundation from which something more formal can be developed. The following sections provide a quick-start guide with worked examples to help the user get up and running with the tool.

The tool essentially uses geographical and depth information from the source data (e.g. a global ocean simulation) and destination simulation (i.e. the proposed regional NEMO model configuration) to determine which source points are required for data extraction. This is done using a kdtree approximate nearest neighbour algorithm. The idea behind this targetted method is that it provides a generic method of interpolation for any flavour of ocean model in order to set up a regional NEMO model configuration. At present (alpha release) the tools do not contain many options, but those that exist are accessed either through a NEMO style namelist or a convient GUI.

PyNEMO has been updated to include integration with CMEMS data repository and the ability to produce tidal boundaries using TPXO or FES2014 as boundary input.

CHAPTER 2

Installation

This page provides a guide to installing PyNEMO.

2.1 Dependencies

anaconda:

- basemap=1.2.0
- netcdf4=1.5.3
- pyqt=5.9.2
- scipy=1.2.1
- python=3.7.6
- pip=20.0.2
- pandas=1.0.1
- pytest=5.3.5
- xarray=0.15.0

pip:

- idna==2.9
- lxml==4.5.0
- pyjnius==1.2.1
- seawater==3.3.4
- thredds-crawler==1.5.4
- motuclient==1.8.4
- sphinx==3.0.2

- sphinx-rtd-theme==0.4.3

2.1.1 How do I install PyNEMO?

Steps to take to install PyNEMO, creating a specific conda virtual environment is highly recommended. click here for more about virtual enviroments

- Install Git (outside scope of this guide)
- Clone PyNEMO repository:

```
$ git clone https://github.com/NOC-MSM/PyNEMO.git
```

- Install Conda, either Anaconda or Miniconda (outside scope of this guide)
- Create conda environment for PyNEMO:

```
$ cd PyNEMO
$ conda env create -f pynemo_37.yml
```

- Activate the new virtual environment:

```
$ source activate pynemo3
```

- Install Jave JDK (outside scope of this guide) and link Java Home to conda environment:

```
$ export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-13.0.2.jdk/Contents/Home
↪# see notes below

(Update needed here)
```

NOTE this link has to be set everytime the enviornment is activated. It can be automated using the .bashrc file or by setting environment hooks in the conda activate and deactivate files.

- Install PyNEMO:

```
$ cd /location/of/pynemo/repo
$ python setup.py build
$ python setup.py install
```

This should result in PyNEMO being installed in the virtual environment, and can be checked by entering:

```
$ pynemo -h
```

Resulting in a help usage prompt:

```
$ usage: pynemo [-g] -s -d <namelist.bdy>
   -g (optional) will open settings editor before extracting the data
   -s <bdy filename> file to use
   -d (optional) will download CMEMS data using provided bdy file
```

The virtual environment can be deactivated to return you to the normal prompt by typing:

```
$ conda deactivate
```

To reactivate, the following needs to be typed:

```
$ source activate pynemo3
```

2.1.2 Jave Home Environment path

The above path for Java Home was valid for a Macbook Pro 2015 with macOS Catalina and Java SDK 13.0.2 however for different java versions, operating systems etc this may be different

The conda environment yaml file has been tested with miniconda 3.7 and found to install the environment correctly.

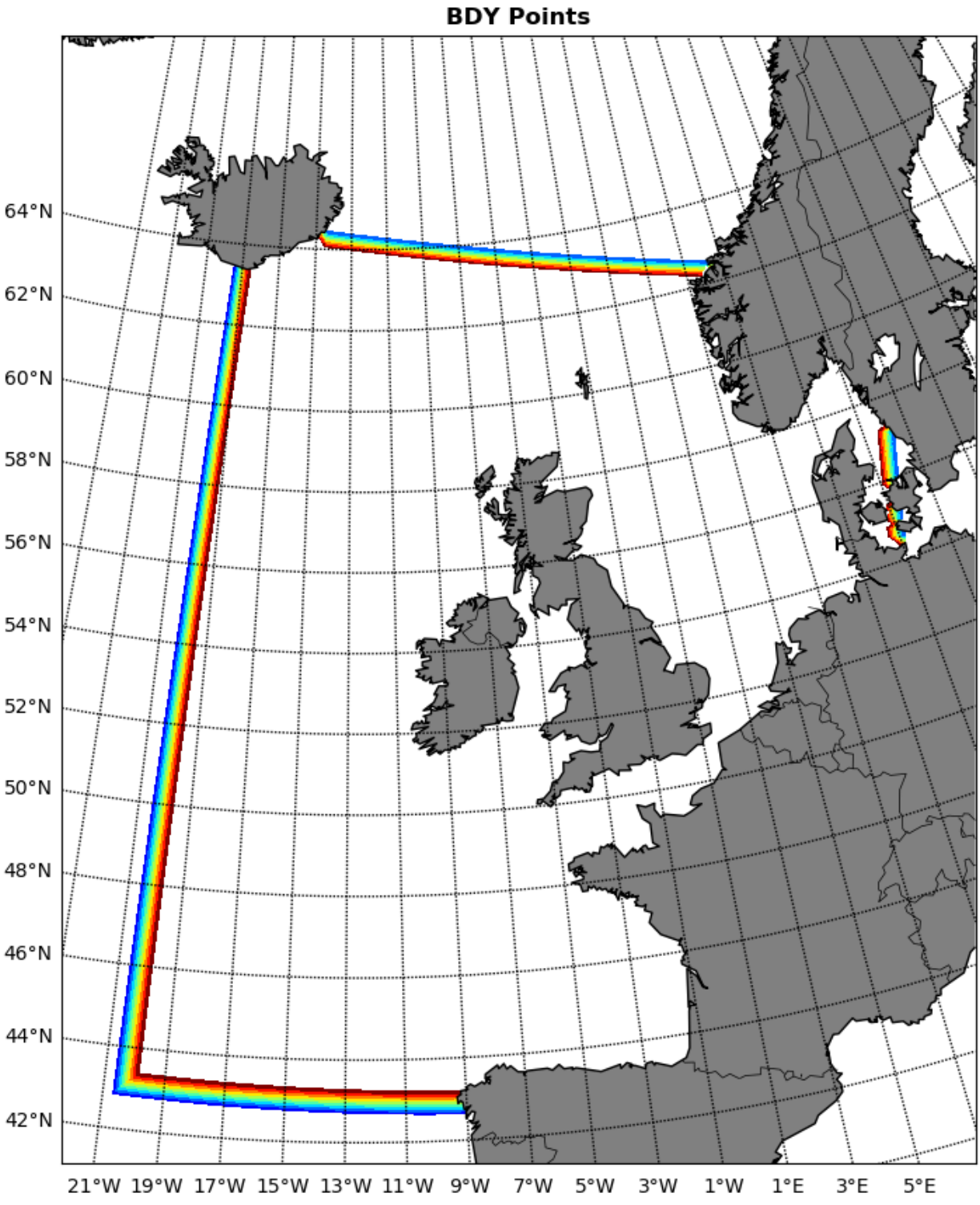
2.1.3 Bench Marking Tests

The PyNEMO module can be tested using the bench marking namelist bdy file in the inputs folder. To check the outputs of the benchmark test, these can be visualised using the plotting script within the test_scripts folder. The following steps are required,

- Run PyNEMO using the namelist file in the inputs folder (namelist_remote.bdy) e.g.:

```
$ pynemo -s /path/to/namelist/file
```

- This will create two output files coordinates.bdy.nc and NNA_R12_bdyT_y1979)m11.nc in an outputs folder
- To check the coordinates.bdy.nc has the correct boundary points, the script bdy_coords_plot.py will plot the domain boundaries and shown the different locations of the rim width (increasing number should go inwards) This script is located in the test_scripts folder.
- The result should look like this (if using the current benchmark data)


BDY Points
64°N
62°N
60°N
58°N
56°N
54°N
52°N
50°N
48°N
46°N
44°N
42°N
21°W
19°W
17°W
15°W
13°W
11°W
9°W
7°W
5°W
3°W
1°W
1°E
3°E
5°E

2.1.4 Unit Tests

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.

Currently **(26/03/2020)** 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 will be increased in the future to cover more PyNEMO functionality.

For more information regarding the use and development of PyNEMO see: [PyNEMO Wiki](https://github.com/jdha/PyNEMO/wiki)

CHAPTER 3

Usage

There are four tools available in pyNEMO. These are "boundary file generation" where boundary data files are generated from a parent grid along the boundary of a child grid. Boundary data can comprise of tracers such as temperature and salinity. Or tidal data from global tide models. Sea surface height and ocean currents are also supported. PyNEMO now has an integrated CMEMS repository downloader. Invoking this option will download data of interest (as specified in NCML file) for a region of interest (as specified in BDY file). PyNEMO uses NCML files to define variable names and data location. this allows multiple netcdf input files to appear as one. This commonmly used on THREDDS servers but is also used locally for CMEMS boundary data input. The GUI allows this NCML files to be generated. Finally there is a settings editor that allows you to edit the pynemo configuration file (BDY file)

3.1 Boundary file generation

This command line tool reads a BDY file, extracts boundary data and prepares the data for a NEMO simulation. The bdy file is a plain text file containing key value pairs. Please look at the sample namelists in the github repository. They are stored in the inputs directory.

PyNEMO now also requires an NCML file (Netcdf markup) that defines the variables and remaps their names so that they are compatable with PyNEMO. This is most commonly required with CMEMS runs as the variable names are different. In previous versions PyNEMO was able to scan a local directory for netcdf files, this is now no longer supported and an NCML file MUST be referenced in the bdy file.

Note: PyNEMO now requires an NCML file as well as a BDY file to run, this can be adapted from the examples in the inputs folder or generated using the NCML GUI.

There are three examples of ncml files and they all use the same child grid but ultilise different data sources. One uses local data, the other uses data hosted on a THREDDS server. The last one is configured to download CMEMS data first and then run using the downloaded data. The namelist file shares common syntax with the NEMO simulation namelist input file.

Note: Directory paths in bdy file can be relative or absolute. The application picks the relative path from the current

working directory.

Syntax for pynemo command is

```
> pynemo [-g] -s <bdy file>
```

For help

```
> pynemo -h
> usage: pynemo [-g] -s -d <namelist.bdy>
>        -g (optional) will open settings editor before extracting the data
>        -s <bdy filename> namelist file to use to generate boundary data
>        -d <bdy filename> namelist file to use to download CMEMS data
```

Example comamnd

```
> pynemo -g -s namelist.bdy
```

3.2 CMEMS data download

To download CMEMS data, the flag -d needs to be specified when running pynemo. This will use the specified namelist file and download the relevent CMEMS data. Once successful the same namelist file can be used to generate the boundary conditions by running PyNEMO with the -s flag. Example command.:

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

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 utils folder needs to be created with two defined strings one called user and the other called pwd to define the user name and password.:

```
$ touch pynemo/utils/CMEMS_cred.py
$ vim pynemo/utils/CMEMS_cred.py
press i
user='username goes here'
pwd='password goes here'
press esc and then :wq
```

IMPORTANT This will create a py file in the right place with the parameters required to download CMEMS, the password is stored as plain text so please do not reuse any existing password you use!

The CMEMS download usage page has more information about how to configure the namelist file.

3.3 GUI NCML Generator

This GUI tool facilitates the creation of a virtual dataset for input into PyNEMO. The virtual dataset is defined using NetCDF Markup Language (NcML).

Using NcML, it is possible to:

- modify metadata
- modify and restructure variables

- combine or aggregate data from multiple datasets. The datasets may reside in the local file system or in a remote OPeNDAP (http://www.opendap.org/) server.

Please see NcML generator usage page for more information in using the GUI.

3.4 pynemo settings editor

This tool will open a window where you can edit the mask and change the values of bdy parameters.

Syntax for pynemo_settings_editor command is

```
> pynemo_settings_editor [-s <bdy filename>]
```

Note: If no file name is specified then a file dialog box will open to select a file.

For help

```
> pynemo_settings_editor -h
> usage: pynemo_settings_editor -s <namelist.bdy>
```

Example:

```
pynemo_settings_editor -s namelist.bdy
```

CHAPTER 4

pyNEMO NcML Generator Usage

This GUI tool facilitates the creation of a virtual dataset for input into pyNEMO. The virtual dataset is defined using NetCDF Markup Language (NcML).

Using NcML, it is possible to:

1. modify metadata
2. modify and restructure variables
3. combine or aggregate data from multiple datasets. The datasets may reside in the local file system or in a remote OPeNDAP (http://www.opendap.org/) server.

4.1 Generator GUI

Users need to follow three distinct steps when using the GUI to generate the virtual dataset:

1. Define a target output NcML file
2. Define the individal variable
3. Generate the NcML file

4.1.1 Define a Target Output File

User should provide the path and name of the target NcML file. The convention is to use *.ncml* as the file suffix. The target file can be specified manually using the input text box or visually using the *Select file* button. Clicking the button will bring up a file dialogue.

4.1.2 Define the Individual Data Variable

The nemo data variables are grouped into the following types :

1. Tracer (temperature, salinity)

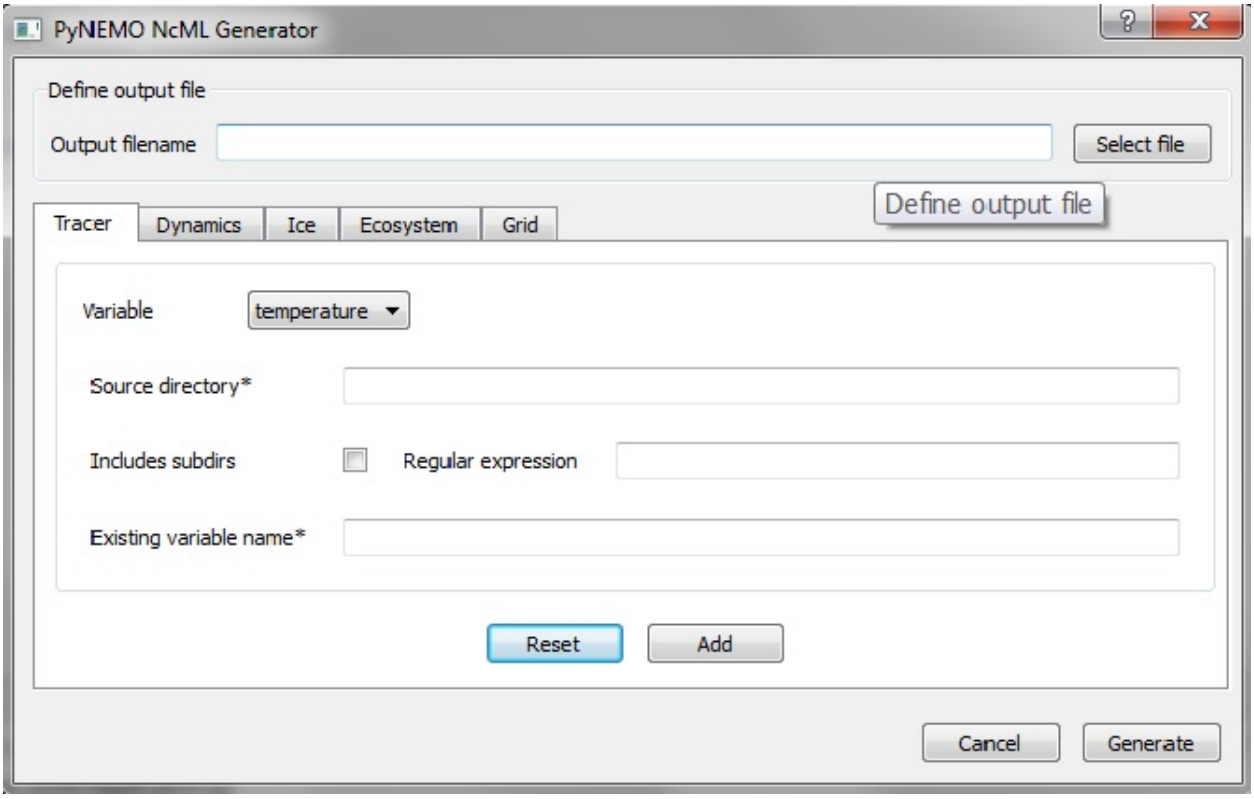


Fig. 1: Overview of the NcML Generator GUI.

2. Dynamics (zonal velocity, meridian velocity, sea surface height)
3. Ice (ice thickness, leads fraction, snow thickness)
4. Ecosystem (reserved for future use)
5. Grid (reserved for future use)

Users can access the required variable by selecting the tab widget and the variable from the *Variable* dropdown list.

For each variable, users must provide information for:

- Source directory - the location of the folder containing the input datasets. User can provide an absolute path to a local file folder or an OPeNDAP endpoint, e.g. http://esurgeod.noc.soton.ac.uk:8080/thredds/dodsC/PyNEMO/data/
- Existing variable name - name used in the source datasets

Users may further filter the source datasets using:

- Include subdirs - check the box to include contents in the sub directories under the specified *Source directory*
- Regular expression - provides a search pattern for filtering the files. See the **Regex** section below for more information.

After completing the variable form, users should click the *Add* button to store the input value. Alternatively, users can use the *Reset* button to reset the input to the previously saved values. If there are no existing values, the variable tab will be reset to the default state.

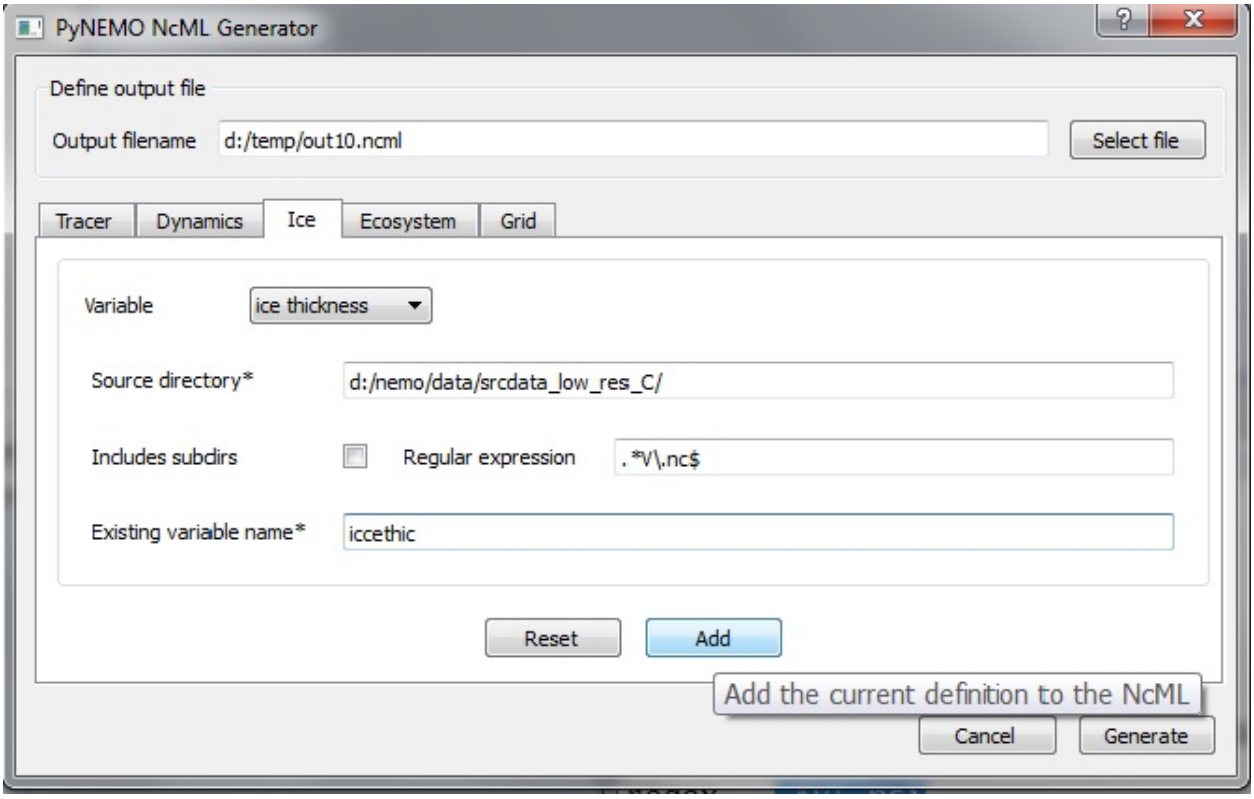


Fig. 2: Example definition of the *Ice thickness variable*.

4.1.3 Generate the NcML file

After adding all the variables, users can generate the NcML file by clicking the *Generate* button. If the operation is successful, a pop-up confirmation dialogue will appear. The generated NcML file can then be used in the bdy file to set up the NEMO simulation.

4.2 Regular Expression (Regex)

Regular expression is a special text string for describing a search pattern to match against some text. You may compare using regex to filter what files to include in your datasets against using wildcard (*) to specify a file search pattern in your computer.

A detailed description of how to define regular expression for filtering datasets in NcML is available at http://www.unidata.ucar.edu/software/thredds/current/netcdf-java/ncml/AnnotatedSchema4.html#regexp.

The following table provides some typical examples of regex:

Regex	Matching File Path	Description
.*V.nc$	c:/dir/dir/dir/abcV.nc	The file path ends in
	d:/muV.nc	V.nc
.*.nc$	c:/dir/dir/dir/*.nc	The file suffix is nc
	d:/*.nc	
.*/2015.*.nc$	c:/dir/2015_01_16.nc	The file path contains
	d:/2015*.nc	2015 and the file suffix
	e:/a/b/c/20151106T.nc	is nc

CHAPTER 5

Tidal Boundary Conditions Generation

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

5.1 Namelist options

To use the namelist needs to be configured with the required options. These are listed below:

```
ln_tide        = .true.              !  =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)      = 'O1'
clname(4)      = 'K1'
clname(5)      = 'N2'
ln_trans       = .false.               !  interpolate transport rather than velocities
! TPXO file locations
sn_tide_grid   = './grid_tpxo7.2.nc'
sn_tide_h      = './h_tpxo7.2.nc'
sn_tide_u      = './u_tpxo7.2.nc'
! location of FES data
sn_tide_fes      = './FES/'
```

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 anyname 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.

5.2 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 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.

5.3 Tide Checker Example Output for M2 U currents

	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

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 ampltiude 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.

5.4 Amplitude Threshold

Important: Percentage Exceedance = 26.933 * Reference Amplitude ^ -0.396'

5.5 Phases Threshold

Important: Phase Exceedance = 5.052 * PyNEMO Amplitude ^ -0.60

5.6 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.

CHAPTER 6

CMEMS downloader usage

IMPORTANT The CMEMS downloader has only been tested with the GLOBAL_ANALYSIS_FORECAST_PHY_001_024 model and specifcally the hourly SSH and U V product. This also has temperature stored within it, but not salinity. Other models and products should work but are currently likely to need some changes to the code to cope with different variable names within the data. This will be fixed in a later release of PyNEMO that is able to handle different variable and tracer names.

PyNEMO has a CMEMS downloading function incorporated within it, this will download a section of the CMEMS global model (more models to be added) '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 below:

```
!-------------------------------------------------------------------------------
!  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
```

(continues on next page)

(continued from previous page)

```
!-------------------------------------------------------------------------------
   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
```

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.

6.1 I/O and NCML file

The location of the NCML file is listed a string defining the source directory or "sn_src_dir". The output folder is also defined here as "sn_dst_dir", **NOTE** if this directory does not exist it will need to be created and permissoned correctly for PyNEMO to run properly. The NCML file details the input files to agregate and what the variable names are. This file can be generated using the ncml_generator, with variable names found using the CMEMS catalogue. https://resources.marine.copernicus.eu/?option=com_csw&task=results 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.

Firstly, 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.

6.2 Data Source Configuration

The first section defines the CMESM data source configuration. The boolean "ln_use_cmems" when set to true will use the CMEMS downloader function to download the requested data, this is defined in the ncml file which can be generated using the NCML generator. Among other things this file defines what data variables to download. This term also changes the variable names to CMEMS specific ones e.g. thetao for temperature and so for salinity. This is in contrast to the NEMO specific ones such as Votemper and Vosaline. When set to false no download occurs and variable names are kept to NEMO specific.

6.3 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 saliniy. 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 relevent size. The booleans "ln_downlad_static" and "ln_subset_static" determine this behavior. Finally there is an int named "nn_num_retry" this defines the number of times to retry downloading the CMEMS data. The data connections are prone to failure so if a non critical 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 efficent 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://resources.marine.copernicus.eu/?option=com_csw&task=results Currently PyNEMO has only been tested using the physical global forecast model although the downloader should be able to download other models and products, it has not been tested and their are known issues with other products that restrict seamless download. e.g. the NorthWest Atlantic model is not currently compatible due to differences in how the model variables are stored.

6.4 FTP Configuration for Static and Grid files

The next section CMEMS FTP configuration, defines which FTP server, remote directory and files to download. This should require modification unless CMEMS changes the file structure or names. 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
```

6.5 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.

CHAPTER 7

Examples

Here we provide two worked examples using pyNEMO. The first is a setup of the Northwest European Shelf using a remote dataset. The second is an end-to-end setup of a small regional model in the tropics.

CHAPTER 8

Example 1: Northwest European Shelf

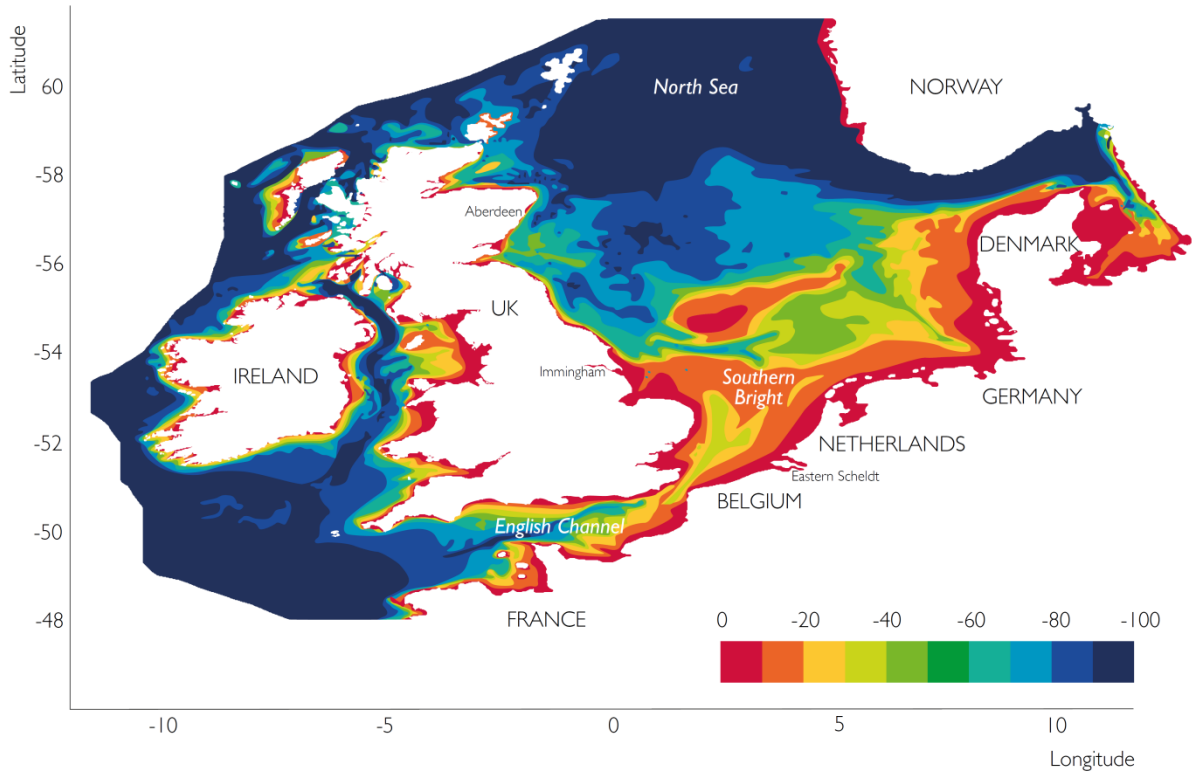


Fig. 1: Northwest European Shelf Bathymetry

This example has been tested on the ARCHER HPC facillity *(22 Feb 2017)*.

First, create a working directory into which the code can run. All the data required for this example are held on a THREDDS server so no addtional data are required.

Note: make sure cray-netcdf-hdf5parallel cray-hdf5-parallel are loaded. This example has been consructed under PrgEnv-intel. e.g.

```
module swap PrgEnv-cray PrgEnv-intel
module load cray-netcdf-hdf5parallel
module load cray-hdf5-parallel
```

Note: Be careful to avoid symbolic links in NEMO control files.

```
cd $WDIR
mkdir OUTPUT
```

Now we're ready to generate the boundary conditions using pyNEMO. If this is not installed follow the *installation guide* or a quick setup could be as follows:

```
cd ~
module load anaconda
conda create --name pynemo_env scipy=0.16.0 numpy matplotlib=1.5.1 basemap netcdf4␣
↪libgfortran=1.0.0
source activate pynemo_env
conda install -c conda-forge seawater=3.3.4
conda install -c https://conda.anaconda.org/srikanthnagella thredds_crawler
conda install -c https://conda.anaconda.org/srikanthnagella pyjnius
export LD_LIBRARY_PATH=/opt/java/jdk1.7.0_45/jre/lib/amd64/server:$LD_LIBRARY_PATH
svn checkout https://ccpforge.cse.rl.ac.uk/svn/pynemo
cd pynemo/trunk/Python
python setup.py build
export PYTHONPATH=~/.conda/envs/pynemo/lib/python2.7/site-packages/:$PYTHONPATH
python setup.py install --prefix ~/.conda/envs/pynemo
cp data/namelist.bdy $WDIR
cd $WDIR
```

Next we need to modify the namelist.bdy file to point it to the correct data sources. First we need to create an ncml file to gather input data and map variable names. First we update *sn_src_dir*, *sn_dst_dir* and *cn_mask_file* to reflect the working path (e.g. sn_src_dir = '$WDIR/test.ncml', sn_dst_dir = '$WDIR/OUTPUT' and cn_mask_file = '$WDIR/mask.nc'). Explicitly write out $WDIR. Next we need to generate test.ncml.

Note: pynemo may have to be run on either espp1 or espp2 (e.g. ssh -Y espp1) as the JVM doesn't have sufficient memory on the login nodes.

```
ssh -Y espp1
module load anaconda
source activate pynemo_env
cd $WDIR
pynemo_ncml_generator
```

For each of the tracer and dynamics variables enter the following URL as the source directory:

http://esurgeod.noc.soton.ac.uk:8080/thredds/dodsC/PyNEMO/data

Add a regular expression for each (Temperature, Salinity and Sea Surface Height each use: .*T.nc$ and the velocities use .*V.nc$ and .*V.nc$) After each entry click the Add button. Finally fill in the output file including directory path

(this should match *sn_src_dir*). Once this is complete click on the generate button and an ncml file should be written to $WDIR.

Then using pynemo we define the area we want to model and generate some boundary conditions:

Note: I've had to add the conda env path to the $PYTHONPATH as python does seem to be able to pick up pyjnius!?

```
export LD_LIBRARY_PATH=/opt/java/jdk1.7.0_45/jre/lib/amd64/server:$LD_LIBRARY_PATH
export PYTHONPATH=~/.conda/envs/pynemo_env/lib/python2.7/site-packages:$PYTHONPATH
pynemo -g -s namelist.bdy
```

Once the area of interest is selected and the close button is clicked, open boundary data should be generated in $WDIR/OUTPUT.

CHAPTER 9

Example 2: Lighthouse Reef

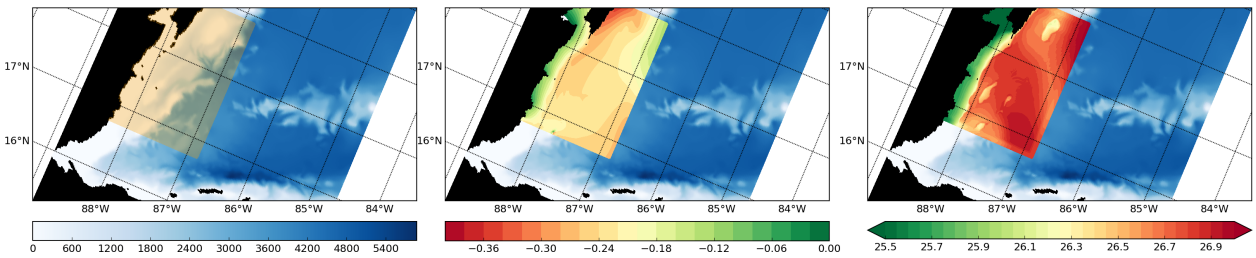


Fig. 1: Regional Mask / SSH after 1 day / SST after 1 day

This example has been tested on the ARCHER HPC facillity.

First, create a working directory into which the NEMO source code can be checked out. Create an inputs directory to unpack the forcing tar ball.

Note: make sure cray-netcdf-hdf5parallel cray-hdf5-parallel are loaded. This example has been consructed under PrgEnv-intel.

```
cd $WDIR
mkdir INPUTS
cd INPUTS
wget ftp.nerc-liv.ac.uk:/pub/general/jdha/inputs.tar.gz
tar xvfz inputs.tar.gz
rm inputs.tar.gz
cd ../
svn co http://forge.ipsl.jussieu.fr/nemo/svn/branches/2014/dev_r4621_NOC4_BDY_VERT_
↪INTERP@5709
svn co http://forge.ipsl.jussieu.fr/ioserver/svn/XIOS/branchs/xios-1.0@629
cd xios-1.0
cp $WDIR/INPUTS/arch-XC30_ARCHER.* ./arch
./make_xios --full --prod --arch XC30_ARCHER --netcdf_lib netcdf4_par
```

Next we setup our experiment directory and drop an updated dtatsd.F90 into MY_SRC to allow the vertical interpolation of initial conditions on to the new verictal coordinates. We also apply several patches for bugs in the code.

Note: when executing ./makenemo for the first time only choose OPA_SRC. For some reason even though LIM_2 is not chosen key_lim2 is in the cpp keys. This means the first call to ./makenemo will fail. Just vi LH_REEF/cpp_LH_REEF.fcm and remove key_lim2 and re-issue the make command.

```
export CDIR=$WDIR/dev_r4621_NOC4_BDY_VERT_INTERP/NEMOGCM/CONFIG
export TDIR=$WDIR/dev_r4621_NOC4_BDY_VERT_INTERP/NEMOGCM/TOOLS
cd $CDIR/../NEMO/OPA_SRC/SBC
patch -b < $WDIR/INPUTS/fldread.patch
cd ../DOM
patch -b < $WDIR/INPUTS/dommsk.patch
cd ../BDY
patch -b < $WDIR/INPUTS/bdyini.patch
cd $CDIR
rm $CDIR/../NEMO/OPA_SRC/TRD/trdmod.F90
cp $WDIR/INPUTS/arch-* ../ARCH
./makenemo -n LH_REEF -m XC_ARCHER_INTEL -j 10
cp $WDIR/INPUTS/cpp_LH_REEF.fcm ./LH_REEF
cp $WDIR/INPUTS/dtatsd.F90 LH_REEF/MY_SRC/
```

To generate bathymetry, initial conditions and grid information we first need to compile some of the NEMO TOOLS (after a small bugfix - and to allow direct passing of arguments). For some reason GRIDGEN doesn't like INTEL:

```
cd $WDIR/dev_r4621_NOC4_BDY_VERT_INTERP/NEMOGCM/TOOLS/WEIGHTS/src
patch -b < $WDIR/INPUTS/scripinterp_mod.patch
patch -b < $WDIR/INPUTS/scripinterp.patch
patch -b < $WDIR/INPUTS/scrip.patch
patch -b < $WDIR/INPUTS/scripshape.patch
patch -b < $WDIR/INPUTS/scripgrid.patch
cd ../../
./maketools -n WEIGHTS -m XC_ARCHER_INTEL
./maketools -n REBUILD_NEMO -m XC_ARCHER_INTEL
module unload cray-netcdf-hdf5parallel cray-hdf5-parallel
module swap PrgEnv-intel PrgEnv-cray
module load cray-netcdf cray-hdf5
./maketools -n GRIDGEN -m XC_ARCHER
module swap PrgEnv-cray PrgEnv-intel
export TDIR=$WDIR/dev_r4621_NOC4_BDY_VERT_INTERP/NEMOGCM/TOOLS
```

Note: my standard ARCHER ENV is intel with parallel netcdf you may need to edit accordingly

Back in $WDIR/INPUTS, create a new coordinates file from the existing global 1/12 mesh and refine to 1/84 degree resolution:

```
cd $TDIR/GRIDGEN
cp $WDIR/INPUTS/namelist_R12 ./
ln -s namelist_R12 namelist.input
./create_coordinates.exe
cp 1_coordinates_ORCA_R12.nc $WDIR/INPUTS/coordinates.nc
```

To create the bathymetry we use the gebco dataset. On ARCHER I had to use a non-default nco module for netcdf operations to work. I also had to cut down the gebco data as the SCRIP routines failed for some unknown reason.

```
cd $WDIR/INPUTS
module load nco/4.5.0
ncap2 -s 'where(topo > 0) topo=0' gebco_1_cutdown.nc tmp.nc
ncflint --fix_rec_crd -w -1.0,0.0 tmp.nc tmp.nc gebco_in.nc
rm tmp.nc
module unload nco cray-netcdf cray-hdf5
module load cray-netcdf-hdf5parallel cray-hdf5-parallel
$TDIR/WEIGHTS/scripgrid.exe namelist_reshape_bilin_gebco
$TDIR/WEIGHTS/scrip.exe namelist_reshape_bilin_gebco
$TDIR/WEIGHTS/scripinterp.exe namelist_reshape_bilin_gebco
```

We perform a similar operation to create the initial conditions:

Note: I've put a sosie pre-step in here to flood fill the land. I tried using sosie for 3D intepolation, but not convinced.

```
cd ~
mkdir local
svn co svn://svn.code.sf.net/p/sosie/code/trunk sosie
cd sosie
cp $WDIR/INPUTS/make.macro ./
make
make install
export PATH=~/local/bin:$PATH
cd $WDIR/INPUTS
sosie.x -f initcd_votemper.namelist
sosie.x -f initcd_vosaline.namelist
$TDIR/WEIGHTS/scripgrid.exe namelist_reshape_bilin_initcd_votemper
$TDIR/WEIGHTS/scrip.exe namelist_reshape_bilin_initcd_votemper
$TDIR/WEIGHTS/scripinterp.exe namelist_reshape_bilin_initcd_votemper
$TDIR/WEIGHTS/scripinterp.exe namelist_reshape_bilin_initcd_vosaline
```

Finally we setup weights files for the atmospheric forcing:

```
$TDIR/WEIGHTS/scripgrid.exe namelist_reshape_bilin_atmos
$TDIR/WEIGHTS/scrip.exe namelist_reshape_bilin_atmos
$TDIR/WEIGHTS/scripshape.exe namelist_reshape_bilin_atmos
$TDIR/WEIGHTS/scrip.exe namelist_reshape_bicubic_atmos
$TDIR/WEIGHTS/scripshape.exe namelist_reshape_bicubic_atmos
```

Next step is to create the mesh and mask files that will be used in the generation of the open boundary conditions:

```
cd $CDIR
cp $WDIR/INPUTS/cpp_LH_REEF.fcm LH_REEF/
ln -s $WDIR/INPUTS/bathy_meter.nc $CDIR/LH_REEF/EXP00/bathy_meter.nc
ln -s $WDIR/INPUTS/coordinates.nc $CDIR/LH_REEF/EXP00/coordinates.nc
cp $WDIR/INPUTS/runscript $CDIR/LH_REEF/EXP00
cp $WDIR/INPUTS/namelist_cfg $CDIR/LH_REEF/EXP00/namelist_cfg
cp $WDIR/INPUTS/namelist_ref $CDIR/LH_REEF/EXP00/namelist_ref
./makenemo clean
./makenemo -n LH_REEF -m XC_ARCHER_INTEL -j 10
cd LH_REEF/EXP00
ln -s $WDIR/xios-1.0/bin/xios_server.exe xios_server.exe
qsub -q short runscript
```

If that works, we then need to rebuild the mesh and mask files in to single files for the next step:

```
$TDIR/REBUILD_NEMO/rebuild_nemo -t 24 mesh_zgr 96
$TDIR/REBUILD_NEMO/rebuild_nemo -t 24 mesh_hgr 96
$TDIR/REBUILD_NEMO/rebuild_nemo -t 24 mask 96
mv mesh_zgr.nc mesh_hgr.nc mask.nc $WDIR/INPUTS
rm mesh_* mask_* LH_REEF_0000*
cd $WDIR/INPUTS
```

Now we're ready to generate the boundary conditions using pyNEMO. If this is not installed follow the *installation guide* or a quick setup could be as follows:

```
cd ~
module load anaconda
conda create --name pynemo_env scipy=0.16.0 numpy matplotlib=1.5.1 basemap netcdf4␣
↪libgfortran=1.0.0
source activate pynemo_env
conda install -c conda-forge seawater=3.3.4
conda install -c https://conda.anaconda.org/srikanthnagella thredds_crawler
conda install -c https://conda.anaconda.org/srikanthnagella pyjnius
export LD_LIBRARY_PATH=/opt/java/jdk1.7.0_45/jre/lib/amd64/server:$LD_LIBRARY_PATH
svn checkout https://ccpforge.cse.rl.ac.uk/svn/pynemo
cd pynemo/trunk/Python
python setup.py build
export PYTHONPATH=~/.conda/envs/pynemo/lib/python2.7/site-packages/:$PYTHONPATH
python setup.py install --prefix ~/.conda/envs/pynemo
cd $WDIR/INPUTS
```

Start up pynemo and generate boundary conditions. First we need to create a few ncml files to gather input data and map variable names. Then using pynemo we define the area we want to model:

Note: pynemo may have to be run on either espp1 or espp2 (e.g. ssh -Y espp1) as the JVM doesn't have sufficient memory on the login nodes.

```
ssh -Y espp1
module load anaconda
source activate pynemo_env
cd $WDIR/INPUTS
pynemo_ncml_generator
```

Note: The ncml files already exist in the INPUTS directory. There is no need generate them. It's a little tricky at the momment as the ncml generator doesn't have all the functionality required for this example. Next step is to fire up pynemo. You can change the mask or accept the default by just hitting the close button (that really should say 'build' or 'go' or such like). Also I've had to add the conda env path to the $PYTHONPATH as python does seem to be able to pick up pyjnius!?

```
export LD_LIBRARY_PATH=/opt/java/jdk1.7.0_45/jre/lib/amd64/server:$LD_LIBRARY_PATH
export PYTHONPATH=~/.conda/envs/pynemo_env/lib/python2.7/site-packages:$PYTHONPATH
pynemo -g -s namelist.bdy
```

Let's have a go at running the model after exiting espp1 (after a few variable renamings, due to inconsistencies to be ironed out):

```
exit
cd $WDIR/INPUTS
```

(continues on next page)

(continued from previous page)

```
module unload cray-netcdf-hdf5parallel cray-hdf5-parallel
module load nco/4.5.0
ncrename -v deptht,gdept LH_REEF_bdyT_y1980m01.nc
ncrename -v depthu,gdepu LH_REEF_bdyU_y1980m01.nc
ncrename -v depthv,gdepv LH_REEF_bdyV_y1980m01.nc
module unload nco
module load cray-netcdf-hdf5parallel cray-hdf5-parallel
cd $CDIR/LH_REEF/EXP00
ln -s $WDIR/INPUTS/coordinates.bdy.nc $CDIR/LH_REEF/EXP00/coordinates.bdy.nc
sed -e 's/nn_msh      =    3/nn_msh      =    0/' namelist_cfg > tmp
sed -e 's/nn_itend    =      1/nn_itend    =       1440 /' tmp > namelist_cfg
cp $WDIR/INPUTS/*.xml ./
qsub -q short runscript
```

CHAPTER 10

Troubleshooting

Always check the PyNEMO log file. This is usually saved in the working directory of PyNEMO as nrct.log. It gives helpful information which may help to diagnose issues. E.g. ValueErrors that are result of a THREDDS server being down and unable to provide data files.

1. pyNEMO crashing in MacOSX (Yosemite)?

- Downgrade the scipy package to 0.15

2. How to make pyNEMO to work behind firewall/proxy?

- Set the environment variable http_proxy. eg. in Linux export http_proxy=<proxy-server>:<proxy-port>

3. Getting this error 'Warning: Please make sure pyjnius is installed and jvm.dll/libjvm.so/libjvm.dylib is in the path' ?

- This error is displayed when the application cannot find the java installation on the local machine. please install a java 7.x runtime from http://www.oracle.com/technetwork/java/javase/downloads/jre7-downloads-1880261.html and append the path to the library in the system path. eg. on windows SET PATH="C:\Program Files (x86)\Java\jre1.7\bin\client" on Linux in shell export LD_LIBRARY_PATH=/opt/java/jdk1.7.0_45/jre/lib/amd64/server:$LD_LIBRARY_PATH in osx export DYLD_LIBRARY_PATH=/System/Library/Java/JavaVirtualMachines/jdk1.7.0_09.jdk/Contents/Home/jre/lib/server:$DYLD_LI

4. Pyjinus error? Socket Timeout? JVM issues? This sometimes happend when requesting tracer boundaries such as temp and salinty along with tidal boundaries. Re running PyNEMO usally works.

CHAPTER 11

Indices and tables

- genindex
- modindex
- search