pyNEMO Documentation

Release 0.1

James Harle, Srikanth Nagella, Shirley Crompton

Jun 06, 2023

Contents

1 Introduction **3**

2 Installation **5**
2.1 Dependencies 5
2.2 Anaconda 5
2.3 From Source 6

3 Usage **7**
3.1 pynemo 7
3.2 pynemo_settings_editor 7

4 pyNEMO NcML Generator Usage **9**
4.1 Generator GUI 9
4.1.1 Define a Target Output File 9
4.1.2 Define the Individual Data Variable 9
4.1.3 Generate the NcML file 11
4.2 Regular Expression (Regex) 11

5 Examples **13**

6 Example 1: Northwest European Shelf **15**

7 Example 2: Lighthouse Reef **17**

8 Troubleshooting **23**

9 Indices and tables **25**

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.

CHAPTER 2

Installation

This page provides a guide to installing pyNEMO.

2.1 Dependencies

1. Python 2.7 (Not tested with 3.x)
2. scipy
3. netCDF4-python
4. numpy
5. matplotlib
6. basemap
7. thredds_crawler
8. seawater
9. pyjnius (optional)

2.2 Anaconda

Using conda: pyNEMO supports Win64, OSX and Linux. for other operating systems please build from source.

Note: It is recommended to create a seperate virtual environment for pyNEMO. Please follow the instructions on doing this at http://www.continuum.io/blog/conda

```
conda install -c https://conda.anaconda.org/srikanthnagella pynemo
```

This will install pynemo and its dependencies. This build is generally outdated as development and bug fixes to the source are a regular occurrence. It may be better to install from source until a beta release is available.

2.3 From Source

Installing pyNEMO using other flavours of software or from source. Install all the dependencies and download the source code from svn and install.

```
svn checkout http://ccpforge.cse.rl.ac.uk/svn/pynemo/trunk/Python/
python setup.py install
```

Note: If building from source in the Anaconda environment all dependencies can be installed using conda apart from thredds_crawler and pyjnius which can be installed using the following Anaconda channel:

```
conda install -c https://conda.anaconda.org/srikanthnagella thredds_crawler
conda install -c https://conda.anaconda.org/srikanthnagella pyjnius
```

CHAPTER 3

Usage

There are two tools available in pyNEMO. They are described in detail below.

3.1 pynemo

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 namelist.bdy file, which 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 <namelist.bdy>
>        -g (optional) will open settings editor before extracting the data
>        -s <bdy filename> file to use
```

Example comamnd

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

3.2 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)

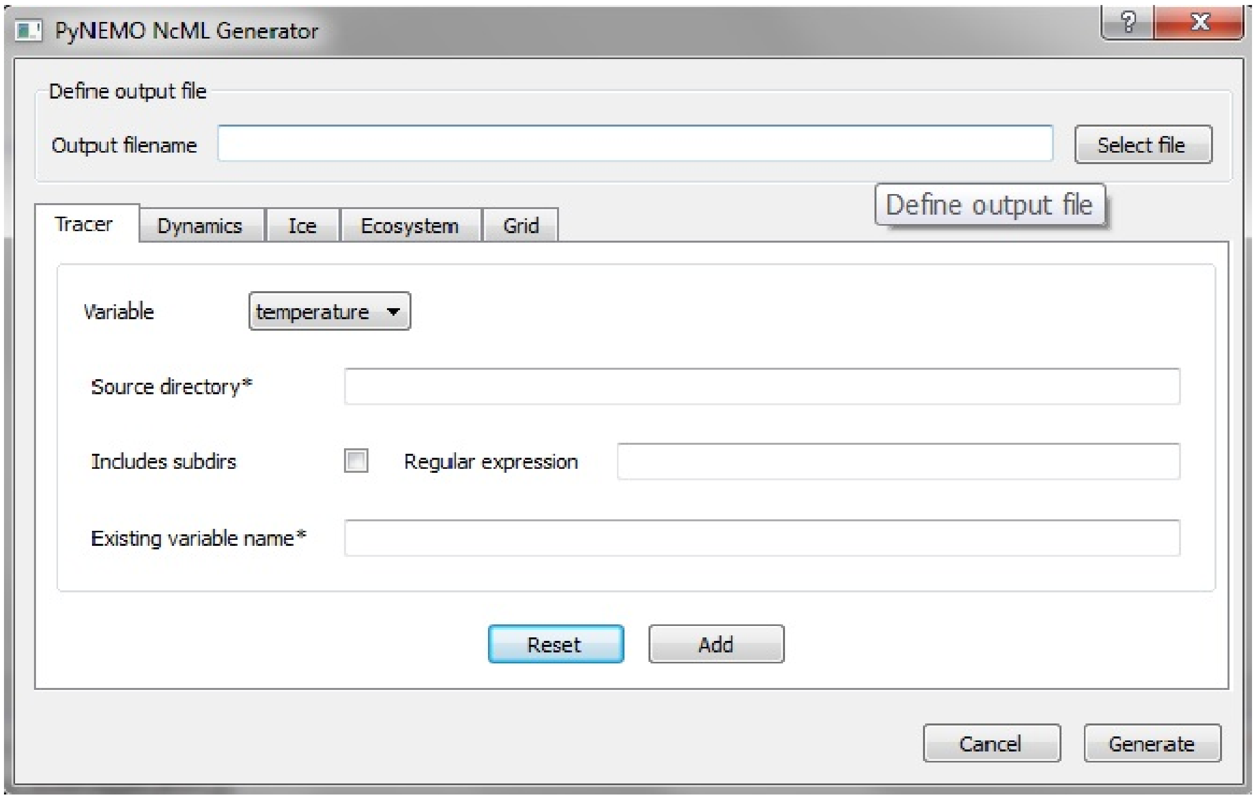


Fig. 1: Overview of the NcML Generator GUI.

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.

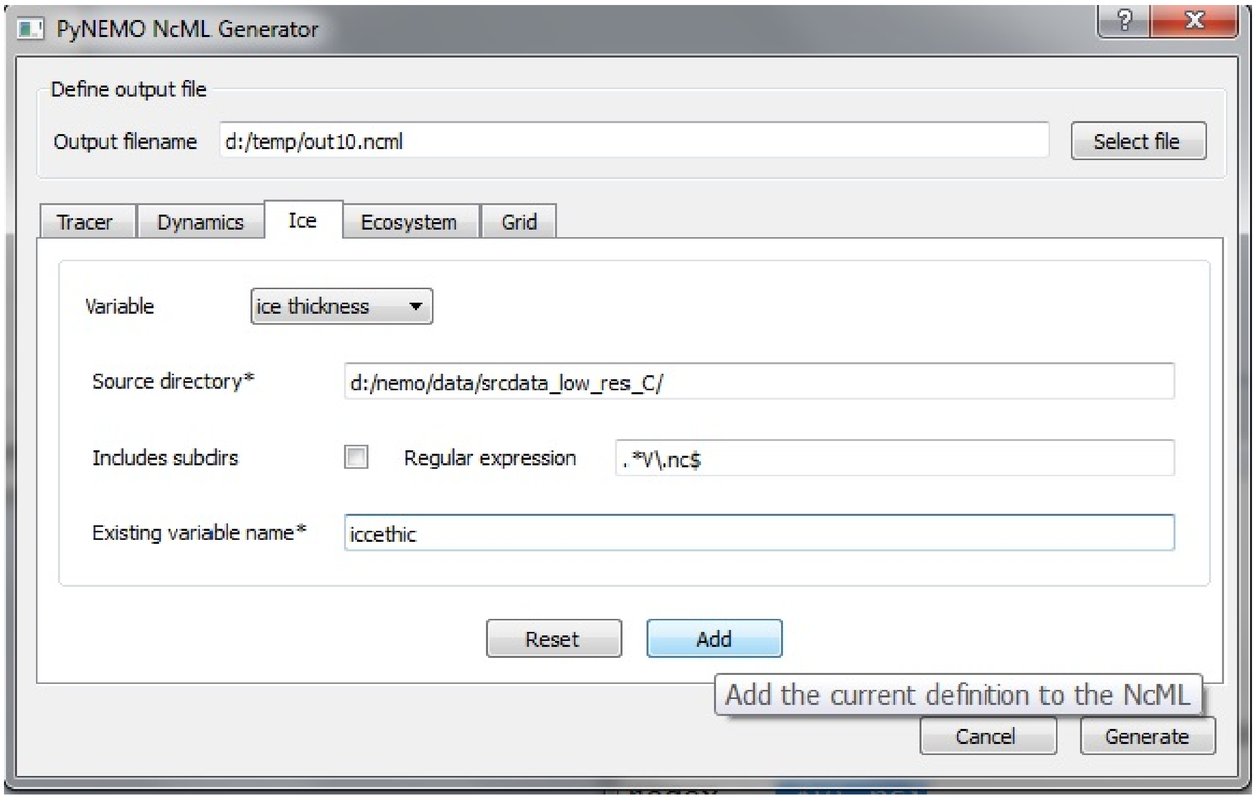
For each variable, users must provide information for:

- 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.



Output filename d:/temp/out10.ncml

Variable ice thickness

Source directory* d:/nemo/data/srcdata_low_res_C/

Includes subdirs ☐ Regular expression .*V\.nc$

Existing variable name* iccethic

Add the current definition to the NcML

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

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 6

Example 1: Northwest European Shelf

_static/eg1.png

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 7

Example 2: Lighthouse Reef

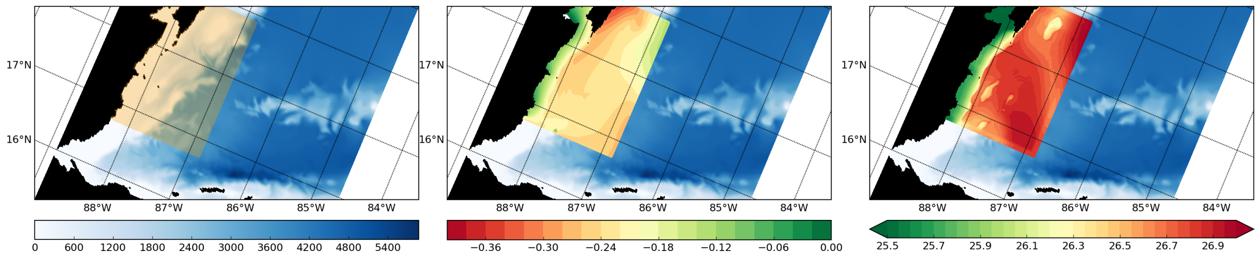


Fig. 1: Regional Mask / SSH after 1 day / SST after 1 day

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 8

Troubleshooting

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

CHAPTER 9

Indices and tables

- genindex
- modindex
- search