NEMO Regional Configuration Toolbox[1]

Harle, J.[1], Nagella, S.[2] and Crompton, S.[3]
[1]National Oceanography Centre, Liverpool.
[2]STFC, Rutherford Appleton Laboratory, Didcot.
[3]STFC, Daresbury Laboratory, Warrington.

Abstract

A toolbox aiding users setting up lateral boundary conditions for a regional NEMO (http://www.nemo-ocean.eu) ocean general circulation model has been developed. The tool is written in Python with portability and sustainability in mind. The NEMO Regional Configuration Toolbox (NRCT) has been developed from existing proprietary *ad-hoc* code improving the flexibility of the tool, especially in term of the IO. An additional feature of the NRCT is to provide access to remote data sets such as the climate model databases from the Climate Modelling Inter-comparison Project (CMIP) 3 and 5[2] that were used in recent Inter-governmental Panel of Climate Change (IPCC) reports. Simple user defined datasets are employed without the need to understand the methods for reading or writing the data. A GUI is also provided to allow the user to define the regional ocean model domain. The NRCT provides an efficient method by which users can setup lateral boundary conditions for near real-time regional NEMO ocean simulations or to begin to perform detailed climate studies with a regional focus with minimal overhead; all of which would be of great scientific benefit and lead to an increased impact of timely scientific output.

1. Introduction

The delineation between coastal and global ocean modelling effort in the UK has been increasingly blurred over the passed few years. The adoption of NEMO (www.nemo-ocean.eu) as a community ocean modelling framework by the Met Office, NERC and many other UKHEs has provided a self-consistent model code with which to simulate the ocean from the global scale down to that of estuaries. As a result the ability to set up regional ocean models within this framework in a tractable manner is becoming increasingly desired within this community. This project will provide a unique set of tools for NEMO users and developers within NERC and UKHEs that currently make use of ARCHER. The toolbox will make progress towards seamlessly setting-up and running forced regional ocean simulations within the NEMO framework, making use of the connectivity and computing power of ARCHER. The toolbox will be configured to access remote data (e.g. via JASIM, CEDA etc.) required to force these regional simulations, thus reducing the need for a multiple step process of transferring and pre-processing data on multiple systems (e.g. in-house clusters or local workstations) before running the final code of ARCHER.

To run a regional ocean model, external data at the edges of the domain are required to provide information about conditions that may influence the simulation e.g. temperatures, currents etc. These data are generally provided from global ocean or climate simulations and are on the whole defined on a coarser resolution model mesh. The core code to be developed and tested will

[1] This work was funded under the embedded CSE programme of the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk)

[2] http://cmip-pcmdi.llnl.gov/cmip5/

involve translating data from an external model source into a format suitable to run a regional NEMO ocean simulation. The key annex to this code will be the ability to access external model data remotely from wherever it may be held using OPeNDAP. Although it is intended that this toolbox is command-line driven, there is also scope to develop a GUI to complete the package.

This project will build on proprietary *ad-hoc* code (in Mathworks MATLAB), largely developed in the NERC eScience GCOMS project (Holt *et al.,* 2009), but here translated to an open-source environment (Python). This will not only benefit those employing it on ARCHER, but also wider NEMO community as a whole (500+ users globally). This toolbox would facilitate users in accessing remote data sets such as the climate model databases from CMIP3 and CMIP5 that were used in recent IPCC reports. This would allow users to begin to perform detailed climate studies with a regional focus or to efficiently set up near real-time simulations with minimal overhead; all of which would be of great scientific benefit and lead to an increased impact of timely scientific output.

2. Software Development

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 kd-tree approximate nearest neighbour algorithm. The idea behind this targeted method is that it provides a generic method of interpolation for any configuration of source ocean model in order to set up a regional NEMO model configuration. Various options are accessed either through a NEMO style namelist or a convenient GUI. The IO abstraction in the toolbox allows the user to simply define where the relevant datasets are held and the variables required without the need to understand the methods used to read or write the data, be they local or held on a remote server.

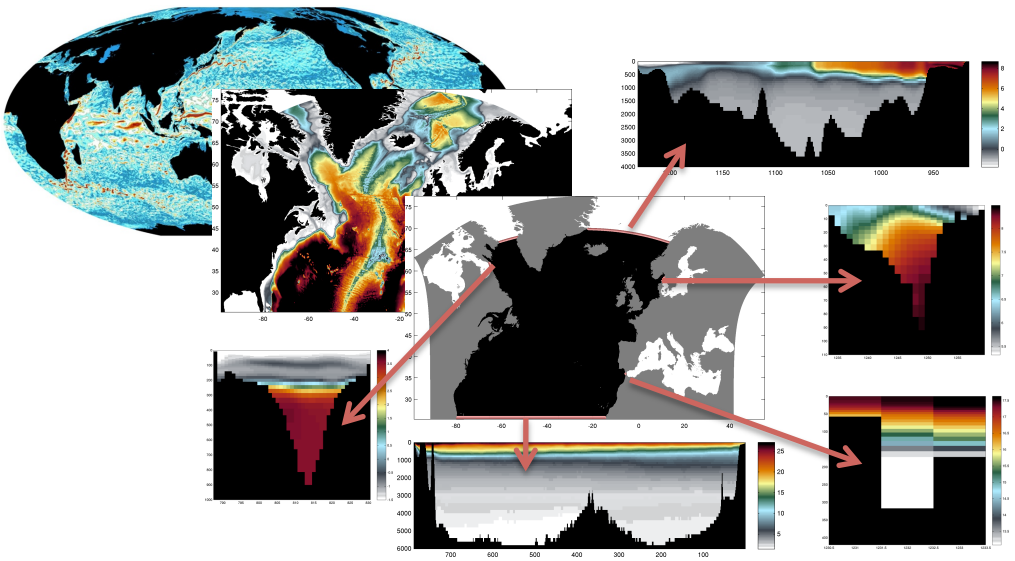

Figure 1: Transfer of information from global simulation to the open boundary conditions of a regional domain.

2.1 Port of proprietary code

The principal work package of the NRCT was to port various pieces of *ad-hoc* code written in Matlab to the open source language of Python. In the process the code was updated and many of the hardwired aspects of the original code were removed. Two example data sets[3] were constructed for benchmarking and performance testing purposes. The data sets were also replicated with non-standard metadata for development and testing outlined in Section 2.2. The code was designed with the Anaconda Python distribution[4] in mind. As this was already available on ARCHER we were able to benefit from the efficient installation of any dependencies required by NRCT. It also allowed for the provision of *conda* builds of the NRCT (see section 3).

Once the code had been ported the NRCT was used to produce open boundary conditions using the example data sets. This was benchmarked against the original Matlab code. Output from both codes produced comparable results with the very minor deviations arising from numerical differences in the language dependant methods used in some of the calculations. The NRCT was run interactively on the Post Processing nodes on ARCHER and on local machines in order to compare the performance of the Python code relative to the Matlab. Both codes performed similarly as a bulk of the processing is devoted to the reading and writing of NetCDF files, which uses very similar protocols in each case. One inefficient found when profiling the Python code with the example data was that around 20% of the overhead was attributable to the initialisation of the *matplotlib* external library. As the example data set is small and only a truncated time period was used in the processing, this overhead would be reduced in *real world* examples. However, we plan to investigate whether this overhead can be minimised in absolute terms. Other approaches to optimisation were also considered, but as yet not been fully tested as were deemed to be outside the scope of the project brief:

- The simplest way of attaining a code speed up would be to implement some form of parallelism. The most obvious and easiest to achieve would be to introduce this is to the time loop in the routines. In general the user will be reading and writing time stamp information over many years, so the time loops within the code could quite easy approach O(100).
- Alternatively, there is the much more complex way of implementing parallelisation by splitting of the spatial grid and doing interpolation in parallel. The advantage of this approach would be that it is less memory intensive than the previous example, which may not been such an issue on the Post Processing nodes of ARCHER, but would be if this code were to be adopted by users to use locally.
- A quick performance gain may also be had by the use of the Python NUMBA[5] package that would allow a degree of speed up around the array

[3] http://esurgeod.noc.soton.ac.uk:8080/thredds/catalog/catalog.html?dataset=PyNEMOtest

[4] https://www.continuum.io

[5] http://numba.pydata.org

oriented code by just-in-time compiling to native machine instruction.

2.2 Remote Data Access

In the proprietary software the IO implementation was rudimentary and limited to a local collection of netCDF files. The second phase of this project was to allow the user to exploit remote data portals using OPeNDAP and to provide increased flexibility in defining data sets. To achieve this, a degree of IO abstraction was implemented with the option to define an entire data set using NcML (netCDF markup language). The use of NcML would effectively wrap a data set up within a descriptor file.

An NcML document is an XML representation of the meta data held within the NetCDF data file. By modifying the NcML file a data set that does not conform to the traditional NEMO format can be used in the NRCT. The other benefit of the NcML is to allow data aggregation given a data directory and search expression. This functionality was only available to local data sets. An alternative method had to be implemented to perform a similar operation on remote data sets. Again the methods employed for the IO were abstracted away as to be no concern to the user.

From the outset we wanted to keep the code base in native Python. Unfortunately when it came to the IO we were presented with a series of issues. While there were various NcML parser routines available in python, many partially written scripts to perform specific tasks, there was nothing like the required functionality of the full Java NetCDF API developed by UNIDATA[6]. So in order to progress the development we exploited a lightweight python package to access java classes (pyjnius[7]). Ideally, we would have preferred to keep everything in native python as this adds a layer of unwanted complexity and overhead, but it was deemed the most appropriate and clean way forward within the time constraints.

To provide similar functionally of the Java NetCDF Tool in terms of local data set aggregation, the Python *thredds crawler* package was adopted to identify the remote data required by the user when running the NRCT. To the user the syntax of interface remains the same, but the abstracted method runs using different subroutines. The example datasets used in the porting of the original code were to be transferred to JASMIN to test the new method of IO within the NRCT. However, there was no THREDDS server available at the time, so an alternative was sought at the National Oceanography Centre, Southampton. The NRCT processing time was of the order 5 times longer when accessing the remotely held data than when accessing the same data locally. This may not immediately appear beneficial, but when considering that some source data sets can be the order of Terabytes in size this method provides a useful and efficient mechanism

[6] http://www.unidata.ucar.edu/software/thredds/current/netcdf-java/

[7] https://pyjnius.readthedocs.org

to access a subset of a larger dataset without the need to transfer the entire data set locally.

The code was then further developed, using the flexibility within the NCML, to provide a wrapper to non-native model output (i.e. the variable naming conventions and meta-data differed to that of NEMO), such as those from CMIP5. One of the original objectives was to demonstrate that access to the CMIP5 database was possible. However the hosting server (Earth System Grid Foundation[8]) was taken down in 2015 due to security concerns and still was not available by the end of the project. The code was successfully tested on one of the existing example data sets on the remote THREDDS server in which all the meta-data had been modified.

A simple Python GUI (Figure 2) was written to provide a suitable interface for the user to define the input data sets required by the NRCT. This allows the user to define the input dataset, without any knowledge of NcML syntax, through a series of tabbed input boxes using variable names, paths and regular expressions. Once the NcML file is generated the NRCT will chose the appropriate IO method depending on whether a local path has been specified. To improve the flexibility of this interface an additional *custom* tab could be added to capture any future user requirements. The idea of using the NcML to define the output file could also be explored, allow the user to not only read in none-native NEMO data, but write out files for none-NEMO simulations.

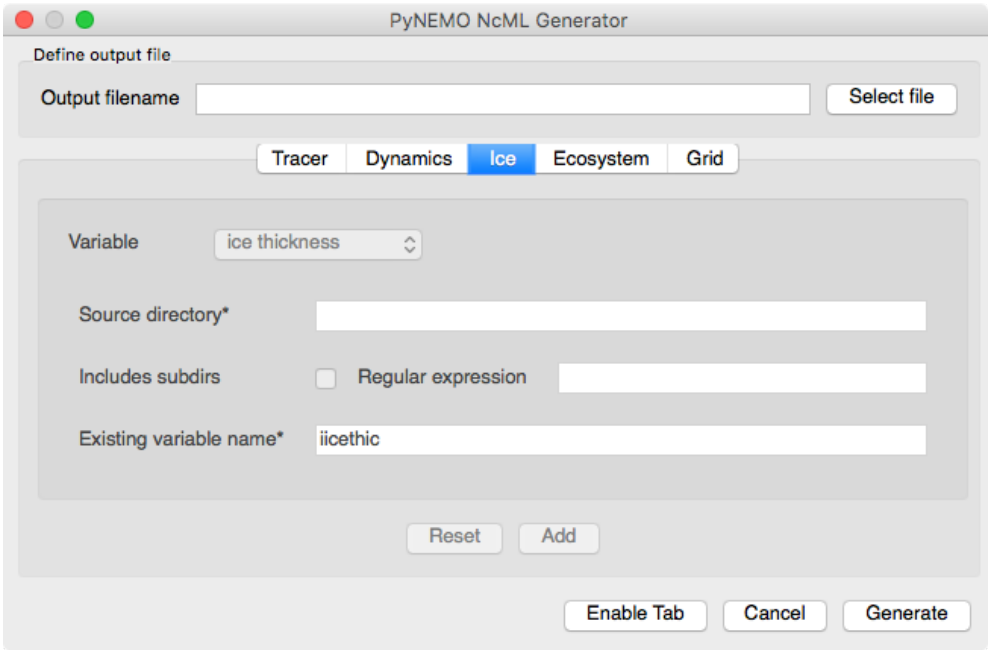

Figure 1: The NcML Generator GUI provides a convenient interface by which the user can generate an XML file describing the data set to be used by the NRCT.

[8] http://esgf.llnl.gov

2.3 NRCT GUI

The NRCT is written to be run as a command line application driven by information provided in a simple namelist file. This assumes that a mask file defining the regional model domain already exists. It is often the case that the user will wish to modify this mask file. In this final work package a GUI to the NRCT was developed to allow the user to define a mask from scratch (Figure 3). This provides user access not only to the namelist, but also the regional domain bathymetry. Using various tool widgets the user can interactively define the region that is to be simulated. Once the GUI is closed the underlying NRCT uses the information provide to generate the open boundary conditions to the selected regional domain. Various methods have been provide for the user to define the regional domain:

- Simple rectangular box selection
- Polygon selection
- The selection of regions shallower than a namelist specified criteria
- The selection of regions shallower than the interface between the coastal/shelf seas and the open ocean

These options are not mutually exclusive and can be used in conjunction with each other allowing the user flexibility to define their domain boundaries.

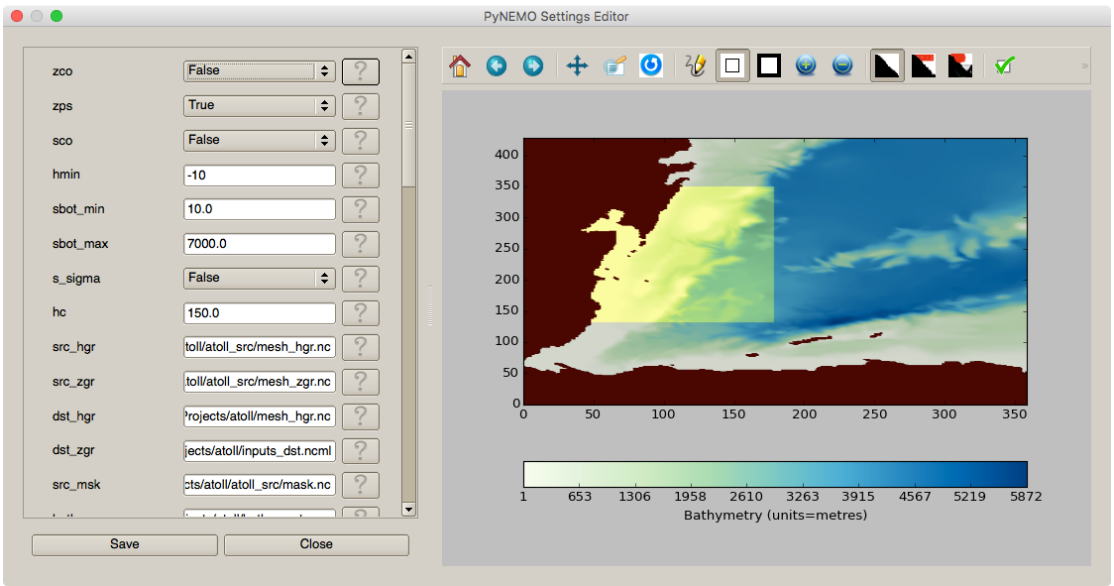

Figure 3: The toolbox GUI allows the users to define the regional ocean domain using a variety of criteria.

The NRCT GUI has been successfully tested interactively on the Post Processing nodes on ARCHER in setting up the following end-to-end example of a NEMO regional simulation:

http://pynemo.readthedocs.io/en/latest/examples.html#example-2-lighthouse-reef

3. Usage

An overview of the software, its installation and examples of usage are provided at: http://pynemo.readthedocs.org/en/latest/intro.html. As it has been written with the Anaconda environment in mind the software can be installed on ARCHER using `conda install` or built from source. At present it is advisable to build from source, as the code is still being actively developed and the builds used in the `conda install` process may be dated.

On ARCHER users can access the toolbox using the Anaconda environment:

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

from source:

```
svn checkout http://ccpforge.cse.rl.ac.uk/svn/pynemo/trunk/Python/
conda install -c https://conda.anaconda.org/srikanthnagella
thredds_crawler
conda install -c https://conda.anaconda.org/srikanthnagella pyjnius
python setup.py install
```

There are builds for Linux, Windows and OSX providing none ARCHER users with access. The alternative is to build from source and as the software is written in Python is accessible by most.

4. Summary

The purpose of the NRCT is to provide an efficient method by which users can setup lateral boundary conditions for near real-time regional NEMO ocean simulations. The toolbox will allow users to begin to perform detailed climate studies with a regional focus with minimal overhead. It is hoped that with further development that this toolbox will be of great scientific benefit and lead to an increased impact of timely scientific output.

An open source toolbox has been produce that can be run with existing open source tools (Python, Java and OPeNDAP) that will benefit NEMO users accessing the ARCHER super computer facility. It will also be made available to the wider NEMO community via the Configuration Manager Workgroup within the NEMO System Team. This will have several scientific benefits:

- The toolbox will facilitate NOC's national capability ocean modelling ability to develop and run high resolution regional forced models.
- Future projects using regional NEMO simulations on ARCHER will greatly benefit from this development (e.g. the recently funded NERC standard grant project: Recicle).
- It will open up/assist the use of IPCC data do perform regional climate studies, an area of great scientific and political interest.
- It will begin to provide tractability/traceability to the process and allow rapid deployment for regional ocean simulations to provide timely

scientific output.

As the code is written in Python it is open source and accessible by the widest possible number of users and developers (Python is being increasingly adopted by the ocean modelling community). There is currently a large NEMO user community (within NERC, UKHEIs and European Institutions) to exploit toolbox and to carry any development forward into the future. In the short to medium term the code will be maintained under National Oceanography Centre National Capability funding, with code and documentation held centrally.

At present the NRCT has been publicised locally at the National Oceanography Centre and at the Met Office and Plymouth Marine Laboratory, with a planned visit to the British Antarctic Survey, Cambridge. Colleagues at the University of Plymouth and CNRS, France have also accessed the tool in the initial testing phase.

Future development:

To complete the NCRT an additional Python module is required to generate the model mesh information for the regional domain. Once this has been written the NCRT will provide and end-to-end method for providing all the data required to perform a NEMO regional ocean simulation.

References

Holt et al. (2009), Modelling the global coastal ocean. Phil. Trans. R. Soc. A, 367, 939–951
doi:10.1098/rsta.2008.0210