NEMO BDY Tools

James Harle
jdha@noc.ac.uk
National Oceanography Centre, Liverpool, UK

15 septembre 2014
– version 1.0 alpha –

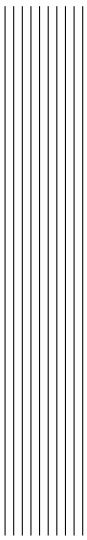

Table des matières

Abstract / Résumé

The ocean engine of NEMO (Nucleus for European Modelling of the Ocean) is a primitive equation model adapted to regional and global ocean circulation problems. It is intended to be a flexible tool for studying the ocean and its interactions with the others components of the earth climate system over a wide range of space and time scales. Prognostic variables are the three-dimensional velocity field, a linear or non-linear sea surface height, the temperature and the salinity. In the horizontal direction, the model uses a curvilinear orthogonal grid and in the vertical direction, a full or partial step z-coordinate, or s-coordinate, or a mixture of the two. The distribution of variables is a three-dimensional Arakawa C-type grid. Various physical choices are available to describe ocean physics, including TKE, GLS and KPP vertical physics. Within NEMO, the ocean is interfaced with a sea-ice model (LIM v2 and v3), passive tracer and biogeochemical models (TOP) and, via the OASIS coupler, with several atmospheric general circulation models. It also support two-way grid embedding via the AGRIF software.

Le moteur océanique de NEMO (Nucleus for European Modelling of the Ocean) est un modèle aux équations primitives de la circulation océanique régionale et globale. Il se veut un outil flexible pour étudier sur un vaste spectre spatiotemporel l'océan et ses interactions avec les autres composantes du système climatique terrestre. Les variables pronostiques sont le champ tridimensionnel de vitesse, une hauteur de la mer linéaire ou non, la temperature et la salinité. La distribution des variables se fait sur une grille C d'Arakawa tridimensionnelle utilisant une coordonnée verticale z à niveaux entiers ou partiels, ou une coordonnée s, ou encore une combinaison des deux. Différents choix sont proposés pour décrire la physique océanique, incluant notamment des physiques verticales TKE, GLS et KPP. A travers l'infrastructure NEMO, l'océan est interfacé avec des modèles de glace de mer, de biogéochimie et de traceurs passifs, et, via le coupleur OASIS, à plusieurs modèles de circulation générale atmosphérique. Il supporte également l'emboîtement interactif de maillages via le logiciel AGRIF.

Disclaimer

Like all components of NEMO, the ocean component is developed under the CECILL license, which is a French adaptation of the GNU GPL (General Public License). Anyone may use it freely for research purposes, and is encouraged to communicate back to the NEMO team its own developments and improvements. The model and the present document have been made available as a service to the community. We cannot certify that the code and its manual are free of errors. Bugs are inevitable and some have undoubtedly survived the testing phase. Users are encouraged to bring them to our attention. The author assumes no responsibility for problems, errors, or incorrect usage of NEMO.

NEMO reference in papers and other publications is as follows :

Madec, G., and the NEMO team, 2008 : NEMO ocean engine. *Note du Pôle de modélisation*, Institut Pierre-Simon Laplace (IPSL), France, No 27, ISSN No 1288-1619.

Additional information can be found on nemo-ocean.eu website.

1 Introduction

Implementing a model configuration that is limited to an oceanic region or a basin is being increasingly desired within the NEMO community. In doing so, the BDY sub-routines can be employed to communicate information from outside the regional model domain into the interior while supporting both outflow and inflow conditions. This set of tools was born out of a requirement to have a generic method by which to provide boundary data for use by these sub-routines.The original code for these tools was written in Mathworks Matlab. It was subsequently translated into Python for wider distribution and to facilitate development. It is far easier to integrate existing scripts into Python, should the need arise.In thier current form these tools are by no means generic and polished, but it is hoped form a foundation from which something more formal can be developed if the desire within the community exists.In the following section there is a summary of usage, along with summary output from two examples that are included with the code.

Pre v3.4 NEMO

Prior to v3.4 NEMO the BDY code relies on time stamp information within the BDY files. For example if a simulation for March 2000 is run with open boundaries supplied with daily mean data, the BDY code requires an input file with 33 time entries : with a corresponding time_counter equal to midday 29 February through to midday 1 April. In v3.4 NEMO time stamp information is discarded with assumed equal time spacing employed. In the example above the BDY input files, if seperated monthly, would consist of three files. One each for February, March and April containing 29, 31 and 30 time entries. However, this becomes an issue if using say 5 day means. For example at the end of March in above example, the last 5 day mean in March may be centred on 28 at 12 :00, thus the BDY takes the first entry in April to complete the boundary forcing data for March, interpolating

in time between the two points. This would imply the 5 day mean in April is centred on 2 at 12 :00. All well and good. However, when the simulation is continued for April the first time entry in the April file is now assumed to be centred on 3 at 12 :00 ($\delta t/2$ into the month where δt is the meaning period of input data. So there can be upto 4 days missmatch in this example. Hence when using source data ¿daily mean ¡ monthly mean all destination BDY files are linearly interpolated onto daily means to aviod this mismatch. Pre v3.4 BDY files are provided as monthly files with an additional time entry for the previous/proceeding month if required. If concatenating pre v3.4 files later, care should be given to the handling of addition time entries to avoid duplications.

Changes between releases

The NEMO BDY tools are current at alpha release and thus far have had no major changes.

• The main modifications are :

1. none as of yet

Future additions ?

Ideas for possible future development of the python tools
• Additions to code :

1. Multiple tide model inputs
2. Handle more than monthly boundary output files
3. Read in existing bdy indices to allow match up with existing simulations
4. B/C grid option
5. Tool to generate NEMO style mesh/mask files to allow use of other model data as input
6. Allow input data to be on a generic vertical grid (z-level only at the moment)
7. netcdf classic output files (currently only v4)
8. have I accounted for EW wrap ?
9. GUI or generic method to provide dst_msk
10. temporal spacing in bdy files
11. convert matlab plot routines to interogate BDY output versus nearest neighbour source file
12. optional output frequency
13. optional temporal smoothing

14. optional spatial smoothin - only 1-2-1 at the moment
15. scale barotropic velocity by src_dep over dst_dep to maintain volume transport

==========
==========

2 Overview of BDY Tools

==========
==========

2.1 Overview

The BDY tools use grid information from the source data (e.g. a global NEMO 025 run) and destination simulation (i.e. the proposed regional simulation) 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 if a NEMO style grid file is produced (tool to be written) for non-*NEMO* source data in principle the BDY tools become more generic. At present (alpha release) the tools do not contain many options, but those that exist at accessed through a NEMO style namelist that is read in when the main python call is made. The following sections summarise the inputs required to produce a set of BDY files to force a regional NEMO simulation.

2.2 Grid Information

```
!------------------------------------------------------------------------
!  grid information
!------------------------------------------------------------------------
   cn_src_hgr = 'some_dir/mesh_hgr_src.nc'
   cn_src_zgr = 'some_dir/mesh_zgr_src.nc'
   cn_dst_hgr = 'some_dir/mesh_hgr_dst.nc'
   cn_dst_zgr = 'some_dir/mesh_zgr_dst.nc'
   cn_src_msk = 'some_dir/mask_src.nc'
```

There is no clean way given a coordinates.nc file and bathymetry.nc file to generate the grid information from the proposed regional simulation, without replicating a lot of the domzgr.F90 code. At this early stage in writing BDY tools

it was deamed cleaner and safer if this information is required as input. This is easily achieved by running the proposed regional simulation for a single timestep with *nn_msh* set to 3 in the *NEMO* namelist file. This will output 3 files : msh_hgr.nc, msh_zgr.nc and mask.nc. A mask from the source grid is also required (*cn_src_msk*). It is often the case the data have be post processed and are missing relavent meta data to determine where the land ocean divide is (e.g. the land values in an ice field are set to zero). The tools rely on the masks and do not search for missing_value or _FillValue properties as these have been found to be unreliable. However, scale_factor and off_set are still read from the source files if they exist and applied to the relevant data.

2.3 Vertical coordinate

```
!------------------------------------------------------------------------
!   vertical coordinate
!------------------------------------------------------------------------
   ln_zco      = .false.   !  z-coordinate - full    steps   (T/F)
   ln_zps      = .true.    !  z-coordinate - partial steps   (T/F)
   ln_sco      = .false.   !  s- or hybrid z-s-coordinate    (T/F)
   rn_hmin     =   -10     !  min depth of the ocean (>0) or
                           !  min number of ocean level (<0)
```

2.3.1 S-coordinate

```
!------------------------------------------------------------------------
!   s-coordinate or hybrid z-s-coordinate
!------------------------------------------------------------------------
   rn_sbot_min =   10.     !  minimum depth of s-bottom surface (>0) (m)
   rn_sbot_max = 7000.     !  maximum depth of s-bottom surface
                           !  (= ocean depth) (>0) (m)
   ln_s_sigma  = .false.   !  hybrid s-sigma coordinates
   rn_hc       =  150.0    !  critical depth with s-sigma
```

2.3.2 Boundary geometry

Each open boundary set is defined as a list of points. The information is stored in the arrays nbi, nbj, and nbr in the idx_bdy structure. The nbi and nbj arrays define the local (i, j) indices of each point in the boundary zone and the nbr array defines the discrete distance from the boundary with $nbr = 1$ meaning that the point is next to the edge of the model domain and $nbr > 1$ showing that the point is increasingly further away from the edge of the model domain. A set of nbi, nbj, and nbr arrays is defined for each of the T, U and V grids. Figure **??** shows an example of an irregular boundary.

The boundary geometry for each set may be defined in a namelist nambdy_index or by reading in a "coordinates.bdy.nc" file. The nambdy_index namelist defines a series of straight-line segments for north, east, south and west boundaries. For the northern boundary, *nbdysegn* gives the number of segments, *jpjnob* gives the j index for each segment and *jpindt* and *jpinft* give the start and end i indices for each segment with similar for the other boundaries. These segments define a list of T grid points along the outermost row of the boundary ($nbr = 1$). The code deduces the U and V points and also the points for $nbr > 1$ if $nn\_rimwidth > 1$.

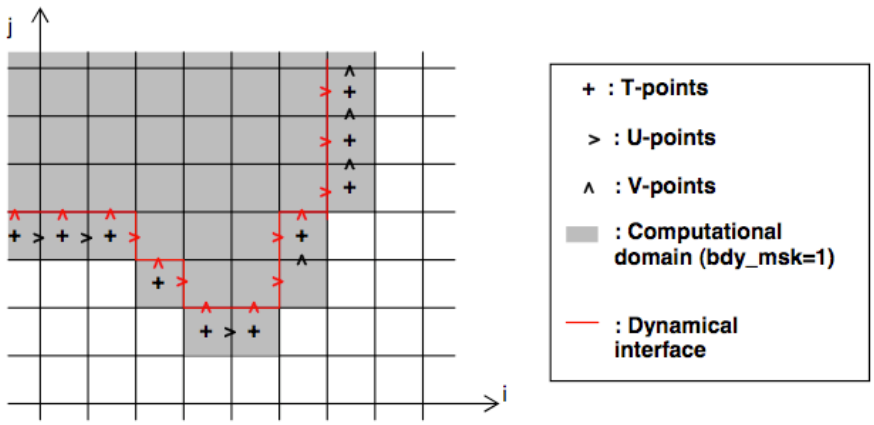

FIGURE 2.1: Example of geometry of unstructured open boundary

The boundary geometry may also be defined from a "coordinates.bdy.nc" file. Figure **??** gives an example of the header information from such a file. The file should contain the index arrays for each of the T, U and V grids. The arrays must be in order of increasing nbr. Note that the nbi, nbj values in the file are global values and are converted to local values in the code. Typically this file will be used to generate external boundary data via interpolation and so will also contain the latitudes and longitudes of each point as shown. However, this is not necessary to run the model.

For some choices of irregular boundary the model domain may contain areas of ocean which are not part of the computational domain. For example if an open boundary is defined along an isobath, say at the shelf break, then the areas of ocean outside of this boundary will need to be masked out. This can be done by reading a mask file defined as *cn_mask_file* in the nam_bdy namelist. Only one mask file is used even if multiple boundary sets are defined.

2.3.3 Input boundary data files

The data files contain the data arrays in the order in which the points are defined in the nbi and nbj arrays. The data arrays are dimensioned on : a time dimension ; xb which is the index of the boundary data point in the horizontal ; and yb which is a degenerate dimension of 1 to enable the file to be read by the standard NEMO I/O routines. The 3D fields also have a depth dimension.

At Version 3.4 there are new restrictions on the order in which the boundary points are defined (and therefore restrictions on the order of the data in the file). In particular :

1. The data points must be in order of increasing nbr, ie. all the $nbr = 1$ points, then all the $nbr = 2$ points etc.

2. All the data for a particular boundary set must be in the same order. (Prior to 3.4 it was possible to define barotropic data in a different order to the data for tracers and baroclinic velocities).

These restrictions mean that data files used with previous versions of the model may not work with version 3.4. A fortran utility *bdy_reorder* exists in the TOOLS directory which will re-order the data in old BDY data files.

```
netcdf med12.obc.coordinates {
dimensions:
      yb = 1 ;
      xbT = 3218 ;
      xbU = 3200 ;
      xbV = 3201 ;
variables:
      int nbit(yb, xbT) ;
      int nbiu(yb, xbU) ;
      int nbiv(yb, xbV) ;
      int nbjt(yb, xbT) ;
      int nbju(yb, xbU) ;
      int nbjv(yb, xbV) ;
      int nbrt(yb, xbT) ;
      int nbru(yb, xbU) ;
      int nbrv(yb, xbV) ;
      float e1t(yb, xbT) ;
            e1t:units = "metres" ;
      float e1u(yb, xbU) ;
            e1u:units = "metres" ;
      float e1v(yb, xbV) ;
            e1v:units = "metres" ;
      float e2t(yb, xbT) ;
            e2t:units = "metres" ;
      float e2u(yb, xbU) ;
            e2u:units = "metres" ;
      float e2v(yb, xbV) ;
            e2v:units = "metres" ;
      float glamt(yb, xbT) ;
            glamt:units = "degrees_east" ;
      float glamu(yb, xbU) ;
            glamu:units = "degrees_east" ;
      float glamv(yb, xbV) ;
            glamv:units = "degrees_east" ;
      float gphit(yb, xbT) ;
            gphit:units = "degrees_north" ;
      float gphiu(yb, xbU) ;
            gphiu:units = "degrees_north" ;
      float gphiv(yb, xbV) ;
            gphiv:units = "degrees_north" ;

// global attributes:
            :file_name = "med12.obc.coordinates.reorder.nc" ;
            :rimwidth = 9 ;
            :NCO = "3.9.9" ;
}
```

FIGURE 2.2: Example of the header for a coordinates.bdy.nc file