Analysis and refactoring of the coupling grids generation procedure in NEMO 3.6

S. Valcke, Andrea Piacentini

CERFACS

Analysis and refactoring of the coupling grids generation procedure in NEMO 3.6

Andrea Piacentini, Sophie Valcke,

07 novembre 2022

CERFACS Technical Report TR-CMGC-22-159

The work reported in this document has been done in the framework of the IS-ENES3 project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824084.

Introduction

When the NEMO ocean global circulation model is coupled to other components of a climate simulation, fluxes and fields are exchanged at the coupling interfaces and interpolated to and from other components grids. Conservative interpolation algorithms need an accurate description of the position of the centres and the corners of each cell of the surface grids at the interface.

The global configurations of NEMO adopt a family of tripolar grids, the ORCA grids, that do not present any singularity at the North Pole, by the use of two meshing poles over the continents. This specificity of the meshing requires particular care in the handling of the periodicity and the folding of the grids.

The algorithms of NEMO do not require the description of the corners of the cells, yet they use a staggered Arakawa C-grid that provides a good starting point for the on-line generation of the input needed by the interpolation libraries. For this reason, a grid generation routine has been included in the extension to the NEMO sources needed for coupled simulations.

The coding of this routine relies on a good comprehension of the ORCA grids definition and of their parallel distribution over the NEMO domain decomposition.

In order to prepare the high-resolution coupling configurations based on the ORCA grid with nominal resolution of $1/12^{th}$ of degree, we have thoroughly analysed the grid generation routine and proposed a partial refactoring to increase its genericity and flexibility.

Grid corners generation

For every coupling interface, the OASIS coupler reads in some NetCDF geometry files with a prescribed format.

Quoting the user guide of the version 5.0 of OASIS3-MCT, section 5.1

> *grids.nc*: contains the model grid longitudes and latitudes in double precision `REAL` arrays. The array names must be composed of a prefix (4 characters), given by the user in the namcouple on the second line of each field (see section 3.3), and of a suffix, `.lon` or `.lat`, for respectively the grid point longitudes or latitudes. If the `SCRIPR/CONSERV` remapping is specified, longitudes and latitudes for the source and target grid **corners** must also be available in the *grids.nc* file as double precision `REAL` arrays dimensioned (`nx`,`ny`,`nc`) where `nc` is the maximum number of corners (in the counterclockwise sense, starting by any corner) over all cells.

> *masks.nc*: contains the masks for all component model grids in `INTEGER` arrays. **Be careful to use the historical OASIS3-MCT convention: 0 = not masked (i.e. active), 1 = masked (i.e. not active) for each grid point.**

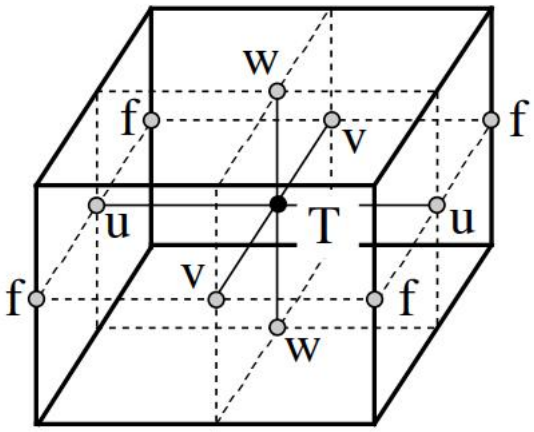
> *areas.nc*: this file contains mesh surfaces for the component model grids in double precision `REAL` arrays.

The NEMO Ocean model does not natively provide all of this information: only the coordinates of the nodes of the finite difference discretization are available, the mask follows a different convention than in OASIS, there are duplicated points at the folding on the North boundary of the grid and for the East-West periodicity, the area of the mesh cells is not immediately available. The `cpl_oasis_gma` routine from the `OPA_SRC/SBC/cpl_oasis3.F90` file computes the needed information and outputs it accordingly to the OASIS3-MCT conventions.

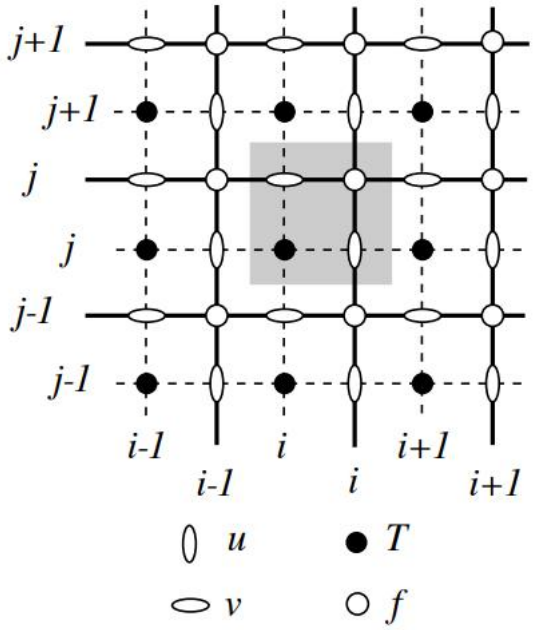
Nemo grids definition

Quoting chapter 4 of the NEMO ocean engine book

> in NEMO, the numerical techniques used to solve the Primitive Equations are based on the traditional, centred second-order finite difference approximation. Special attention has been given to the homogeneity of the solution in the three space directions. The arrangement of variables is the same in all directions. It consists of cells centred on scalar points (t, S, p, ρ) with vector points (u, v, w) defined in the centre of each face of the cells. This is the generalisation to three dimensions of the well-known "C" grid in Arakawa's classification.



In the previous figure, T indicates scalar points where temperature, salinity, density, pressure and horizontal divergence are defined. (u,v,w) indicate vector points, and f indicates vorticity points where both relative and planetary vorticities are defined.
In the model coding, the grid is separated into a vertical coordinate and a horizontal grid stored with the following integer indexing:



Notice that the dashed area indicates the cell in which variables contained in arrays have the same *i*- and *j*-indices.

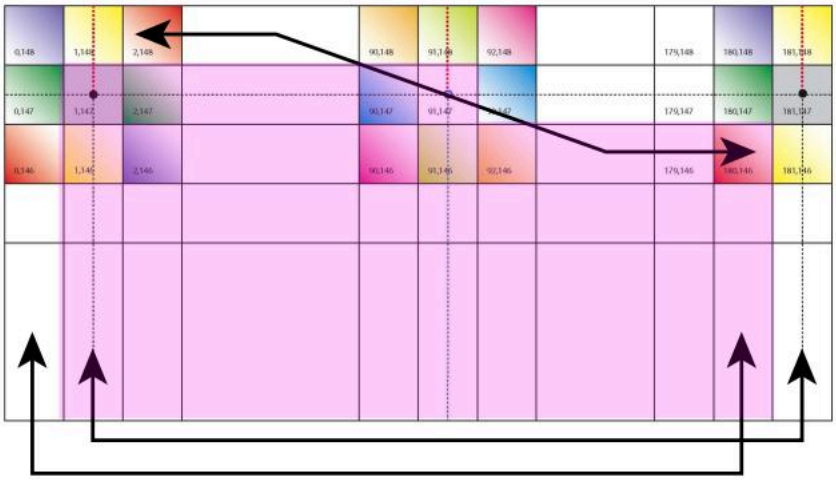
As explained in section 2.3 of the NEMO book, if $\lambda(i,j)$ and $\varphi(i,j)$ designate the discrete longitudes and latitudes on the mesh and if a is the Earth radius and z the altitude above a reference sea level, the local deformation of the horizontal curvilinear coordinate system is given by the scale factors

$$e_1 = (a+z)\left[\left(\frac{\partial\lambda}{\partial i}\cos\varphi\right)^2 + \left(\frac{\partial\varphi}{\partial i}\right)^2\right]^{\frac{1}{2}}$$

$$e_2 = (a+z)\left[\left(\frac{\partial\lambda}{\partial j}\cos\varphi\right)^2 + \left(\frac{\partial\varphi}{\partial j}\right)^2\right]^{\frac{1}{2}}$$

Since the ocean depth is far smaller than the earth's radius, $a + z$, can be replaced by a (thin-shell approximation). The resulting horizontal scale factors e_1, e_2 are independent of the vertical level and the product $e_1 * e_2$ provides a good approximation of the area of the grid cell centred around a grid node.

At the model domain boundaries several choices are offered: closed, cyclic eastwest, south symmetric across the equator, a north-fold, and combination closed north fold or cyclic-north-fold. The north-fold boundary has been introduced in order to handle the north boundary of a three-polar ORCA grid, the one used for the global domain in coupled simulations. Such a grid has two poles in the northern hemisphere and thus requires a specific treatment which, in turns, is specialized accordingly to the grid nominal resolution. The *jperio* parameter triggers the appropriate treatment of the model boundaries. For instance, *jperio* = 4 for the ORCA2 (~2 degrees), ORCA025 (~¼ degree) and ORCA12 (~$^{1}/_{12}$ degree), *jperio* = 6 for the ORCA1 (~1 degree) configuration. In order to handle the cyclic and north-fold boundaries, the mesh indexing is extended to include twice the cells around the open boundaries.



In the previous figure (representing the ORCA 2 grid, with *jperio* = 4), only the pink shaded area corresponds to the inner non duplicated domain.

Parallel decomposition

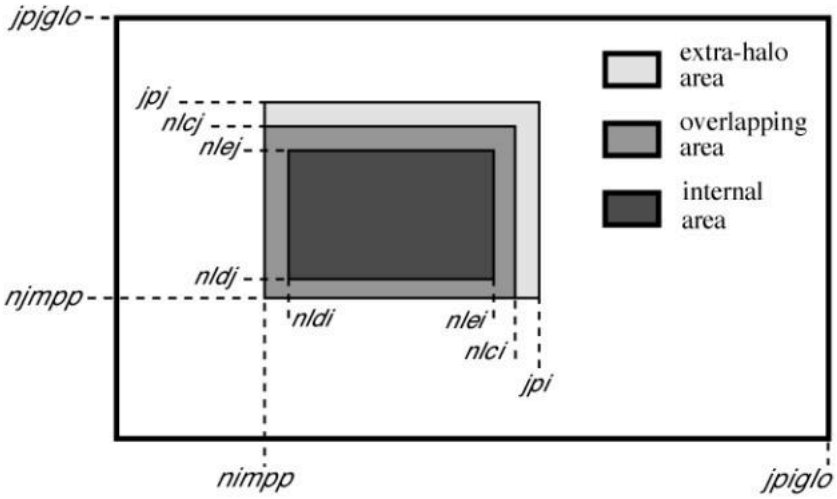
The total number of nodes in the horizontal direction of the mesh is indicated by *jpiglo* and *jpjglo*.

For massively parallel processing (mpp), a domain decomposition method is used. The large computation domain of a numerical experiment is split into several smaller subdomains. Each processor has its own local memory and solves the model equations over a subdomain of the whole model domain. The subdomain boundary conditions are specified through communications between processors which are organized by explicit statements (message passing method) and rely on halos of ghost points around the local subdomain.

The global domain is split into *jpni* subdomains in one direction and *jpnj* in the other. Every processor allocates the same quantity of memory in rectangular objects of overall size (*jpi*, *jpj*), of which the element of local indexing (1,1) corresponds to the (*nimpp*, *njmpp*) position on the global non distributed grid.

Notice that an alternative domain decomposition approach allows for excluding the subdomains that would not contain any ocean point, but only masked land points. The indexing conventions described in the following are valid for both approaches, but the approach without land-only processors will need less than *jpni* * *jpnj* processors at the price of a more complicated communication pattern among active subdomains. The approach without land-only processors is preferred for massively parallel computations, but, as we will see in the following, it is not suitable for generating the grid description.

The number of rows in the halo is usually set to one (*jpreci* = 1 and *jprecj* = 1). A local subdomain includes therefore a local private computation region between indexes (*nldi*, *nlei*) and (*nldj*, *nlej*), halo rows span from 1 to *nldi* and from *nlei* to *nlci* = *nlei* + *jpreci* in the *i* direction and from 1 to *nldj* and from *nlej* to *nlcj* = *nlej* + *jprecj* in the *j* direction. Since the local storage has the same size on all the processors, but the domain decomposition not necessarily divides exactly the global grid dimensions, some processors have an extra non active halo up to the local size.



Because of the numbering conventions and of the treatments of the cyclic or closed boundaries, for the processors lying on the Westmost boundary (*nimpp* = 1) or on the Southmost boundary (*njmpp* = 1) there is no outer overlapping area, hence *nldi* = 1 and *nldj* = 1 respectively. Full details can be found in section 8.3 of the NEMO book. It is important to notice for the following that the output routines only write the content of the local active domain.

The actual domain decomposition and indexing is output at the beginning of a NEMO simulation in the `layout.dat` file.

At every timestep the halos values are updated by a call to the `lbc_lnk` routine from the `OPA_SRC/LBC/lbclnk.F90` module. Unless explicitly excluded with an optional argument, this routine triggers a call to the `lbc_nfd` routine from the `OPA_SRC/LBC/lbcnfd.F90` module for the synchronization of the values around the north-fold boundary.

Nemo grids input

Even if the coordinates of the ORCA grids nodes are obtained from analytical formulae, they are not computed on the fly but read in from an input `coordinates.nc` file.

It provides the following fields for the global non distributed grid (here for the ORCA 1 grid):

```
netcdf coordinates {
dimensions:
      y = 294 ;
      x = 362 ;
variables:
      double e1f(y, x) ;
      double e1t(y, x) ;
      double e1u(y, x) ;
      double e1v(y, x) ;
      double e2f(y, x) ;
      double e2t(y, x) ;
      double e2u(y, x) ;
      double e2v(y, x) ;
      double glamf(y, x) ;
      double glamt(y, x) ;
      double glamu(y, x) ;
      double glamv(y, x) ;
      double gphif(y, x) ;
      double gphit(y, x) ;
      double gphiu(y, x) ;
      double gphiv(y, x) ;
}
```

Where `y` is *jpjglo*, `x` is *jpiglo*, the `e1` and `e2` fields are the scale factors and the `glam` and `gphi` fields are the longitude and the latitude of the nodes for the staggered grid designated by the suffix `f`, `t`, `u` or `v`.

The file is read in by the `hgr_read` routine from the `OPA_SRC/DOM/domhgr.F90` file. For mpp computations, only the values on the active local domain are read in. Then the halos are filled by a call to `lbc_lnk` that does not apply the north-fold boundary treatment. This is important in order to avoid to invert the relative position of the staggered grids when folding the Northmost rows over themselves. Finally, the extra-halo, if present, is completed by replicating the values in the halo up to the size of the local storage.

In this way the grid coordinates and the scale factors are provided on all the *jpi* * *jpj* points of every local domain.

The ocean/land mask in NEMO is computed in the `dom_msk` procedure from the `OPA_SRC/DOM/dommsk.F90` module for every vertical level *k* from the values of the *misfdep* field, which provides the index of the first wet level for every horizontal position. This field is updated when the model bathymetry at T grid locations is read in from the `bathy_meter.nc` file in the `zgr_bat` procedure in the `OPA_SRC/DOM/domzgr.F90` module.

Grid corners

For the description of the coupling interface at the surface, only the NEMO T grid is relevant. Let's indicate in the following the NEMO Ocean Grid at T points by the acronym *nogt*.

We will associate the NEMO `glamt(i,j)` and `gphit(i,j)` fields to the *nogt.lon* and *nogt.lat* fields in the OASIS grids description containing the longitudes and the latitudes of the centres of the grid cells. As we have already stated, these fields are defined on the totality of the local domain storage, and do not need any extra treatment in case of parallel execution of the `cpl_oasis_gma` routine.

In the OASIS grid description, the grid cells corners longitudes and latitudes are stored in the *nogt.clo* and *nogt.cla* fields. Their shape is (*jpiglo*, *jpjglo*, *jpcrn*) where *jpcrn* = 4 is the number of corners per cell.

The OASIS convention states that the corners must be described in counter-clockwise order. We arbitrarily choose to start the description from the corner that lays in the direction of increasing *i* and *j* w.r.t. the centre. Where the grid is not stretched nor folded this corner lays to the North East of the centre, therefore we adopt the following mnemonic numbering of the nodes:

jpne = 1 (North East), *jpnw* = 2 (North West), *jpsw* = 3 (South West), *jpse* = 4 (South East)

We have seen by construction of the staggered grids that the for the cell of global indexes (*i*, *j*) the *jpne* corner coincides with the F grid node of the same indexes. As for the centres, the *jpne* corners are defined on the totality of the local storage.

The *jpnw* corner of the (*i*, *j*) cell coincides with the (*i-1*, *j*) node of the F grid. Analogously, the *jpsw* corner coincides with the (*i-1*, *j-1*) node and the *jpse* corner coincides with the (*i*, *j-1*) node of the F grid. By construction, the *jpnw* and *jpsw* corners are not defined for *i* = 1 (global numbering) and the *jpsw* and *jpse* corners are not defined for *j* = 1 (global numbering).

In order to avoid any possible confusion between local and global numbering, let's recall that the mpp coding of NEMO uses a local storage numbering spanning from (1, 1) to (*jpi*, *jpj*) and that the active region that needs to be filled for a correct output spans from (*nldi*, *nldj*) to (*nlei*, *nlej*). On most of the parallel subdomains, *nldi* = 1+*jpreci* = 2 and *nldj* = 1+*jprecj* = 2, therefore the corners are well defined in the 4 directions. As we already recalled, for the Westmost and the Southmost subdomains *nldi* = 1 and *nldj* = 1 respectively, corresponding to the aforementioned cases of *i* = 1 (global numbering) and *j* = 1 (global numbering). It is only for these subdomains that an action has to be taken for defining the *jpnw*, *jpsw* and *jpse* corners.
For the Westmost subdomains, on the cells with *i* = 1 the cyclic East-West periodicity applies. The coordinates of the *jpnw* and *jpsw* corners can be obtained by periodicity. If the global domain is decomposed in more than one mpp subdomain in the *i* direction, the `lbc_lnk` routine gets the appropriate values by MPI communications with the subdomains on the Eastmost boundary. Notice that in order to avoid reverting the corner numbering on the northfold, the last optional argument of the `lbc_lnk` API has to be set to an arbitrary 3 characters string. If there is a single domain in the *i* direction, the treatment has to be applied by hand, taking into account the lateral boundary conditions: by periodicity if the domain is cyclic in the East-West direction, or by extrapolating Westward the longitudes of the Westmost corners if the domain is closed.
Since the ORCA global grids never extend till the South Pole, the Southmost boundary is always closed, therefore the coordinates of the *jpsw* and *jpse* corners on the Southmost cells are obtained by extrapolating Southward the latitudes of the Southmost corners.

The areas of the cells are approximated by the product of the e_1 and e_2 scale factors on the T grid. Since they are defined on the totality of the local storage, no specific treatment is needed.

Duplicated points removal

As previously explained, the global ORCA grids contain some duplicated cells used in the cyclic and northfold boundary conditions. The *tmask* ocean land mask in NEMO allows for redundant duplicated ocean cells. In OASIS, on the contrary, the cells have to be uniquely defined, therefore the *nogt.msk* in the OASIS `masks.nc` file has to flag as land all but one set of the duplicated cells.
It has to be noticed that the same considerations apply for the output of the *tmaskutil* mask in the mesh/mask files written by the `dom_wri` routine in the `OPA_SRC/DOM/domwri.F90`

module. In this routine, *tmaskutil* is derived from *tmask* by multiplication times an auxiliary mask flagging off the duplicated points and generated by a call to the `dom_uniq` routine.
Making `dom_uniq` a public procedure of the `OPA_SRC/DOM/domwri.F90` module, the auxiliary mask can be generated and used in the `cpl_oasis_gma` routine and the equivalent of *tmaskutil* is stored in *nogt.msk* (see the use of *zuni*).[1]
The auxiliary mask is also used for the cleaning of the Antarctic calving and the run off grids and masks, identified respectively by *noat* and *nort*.

Grids, areas and masks output

Once the coordinates of the cell centres and of the four corners have been computed together with the cells area and the sea/ocean mask without duplicated points and following the OASIS convention (integer values, 0 for active points, 1 for land or duplicated points), they have to be written to the `grids.nc`, `areas.nc`, `masks.nc` OASIS input files.
These are single global files for the whole domain, including the masked land points. It will not be possible, therefore, to output the grid information from a NEMO mpp decomposition excluding the land-only processors.
There are two alternative approaches for the output. The first one relies on the NEMO output procedures from the `iom` module and can be activated by setting the logical *ll_write_iom* to *.TRUE.*. The second one relies on the OASIS grids writing routines described in section 2.2.4 of the OASIS user guide and can be activated by setting the logical *ll_write_iom* to *.FALSE.*.

IOM

With the `iom` procedures, one NetCDF file is written per subdomain. Only the content of the active interior domain with indexes spanning from (*nldi*, *nldj*) to (*nlei*, *nlej*) is written to file, therefore the global domain can be reconstructed simply by tiling the content of the partial files. The `rebuild_nemo` tool from `NEMOGCM/TOOLS` can very effectively recombine the partial files from the subdomains into a single global file. An even faster alternative is the `mppcomb.exe` tool that Kristian Mogensen coded for the NEMOVAR data assimilation suite.
It has to be noted that there is a current limitation in the `iom` procedures of 3D fields: they can only use the number of vertical model levels as a third dimension. The *nogt.clo* and *nogt.cla*

[1] This removes the need to define and use *tpol,* which wrong deallocation in *wrk_alloc* was making the ocean model to abort before calling *oasis_enddef,* preventing the effective writing of the grids, masks and areas in files *grids.nc, masks.nc and areas.nc,* see below.

fields are three-dimensional but only have *jpcrn* = 4 entries in the third dimension and have, for this reason, to be written as four separate 2D fields each. The OASIS compliant fields are reconstructed with a simple post-processing procedure coded in python using the `netCDF4` package and the `numpy.vstack` primitive, `rebuild_corners.py`, or with NCO. The `transform_oasis_grid_files_gen.ncl` script deals with the stacking too.

OASIS

Section 2.2.4 of the OASIS3-MCT User Guide, details the API of these routines:

`oasis_start_grids_writing`

`oasis_write_grid`

`oasis_write_corner`

`oasis_write_mask`

`oasis_write_area`

`oasis_terminate_grids_writing`

Quoting the document:

> If a grid data files does not exist, the corresponding routine will create it; if the grid data file exists, the routine can be used to **add** grid definition fields but it will not **overwrite** grid definition fields already existing in the file with the same grid name.

Care has to be paid in the handling of the ECLIS dataflow, checking that no previous versions of the grid files containing the description of the ocean grid is found in the run directory.

The user guide also states that:

> The creation of the different grid data files is completed in the routine `oasis_enddef`.

Any error in the execution previous of the completion of `oasis_enddef` will imply a failure in the grid generation procedure.

Unfortunately, in the current version of CNRM NEMO 3.6 stable sources adapted to coupled simulations, the NEMO domain decomposition initialization provides a correct initialization of XIOS only for the parallel decomposition neglecting the land-only subdomains. As we've already mentioned, the grid generation procedure must run on a complete decomposition including land-only subdomains, therefore XIOS is not correctly initialized and causes a code crash before the completion of `oasis_enddef`.

Temporary fix in the xios calls from NEMO

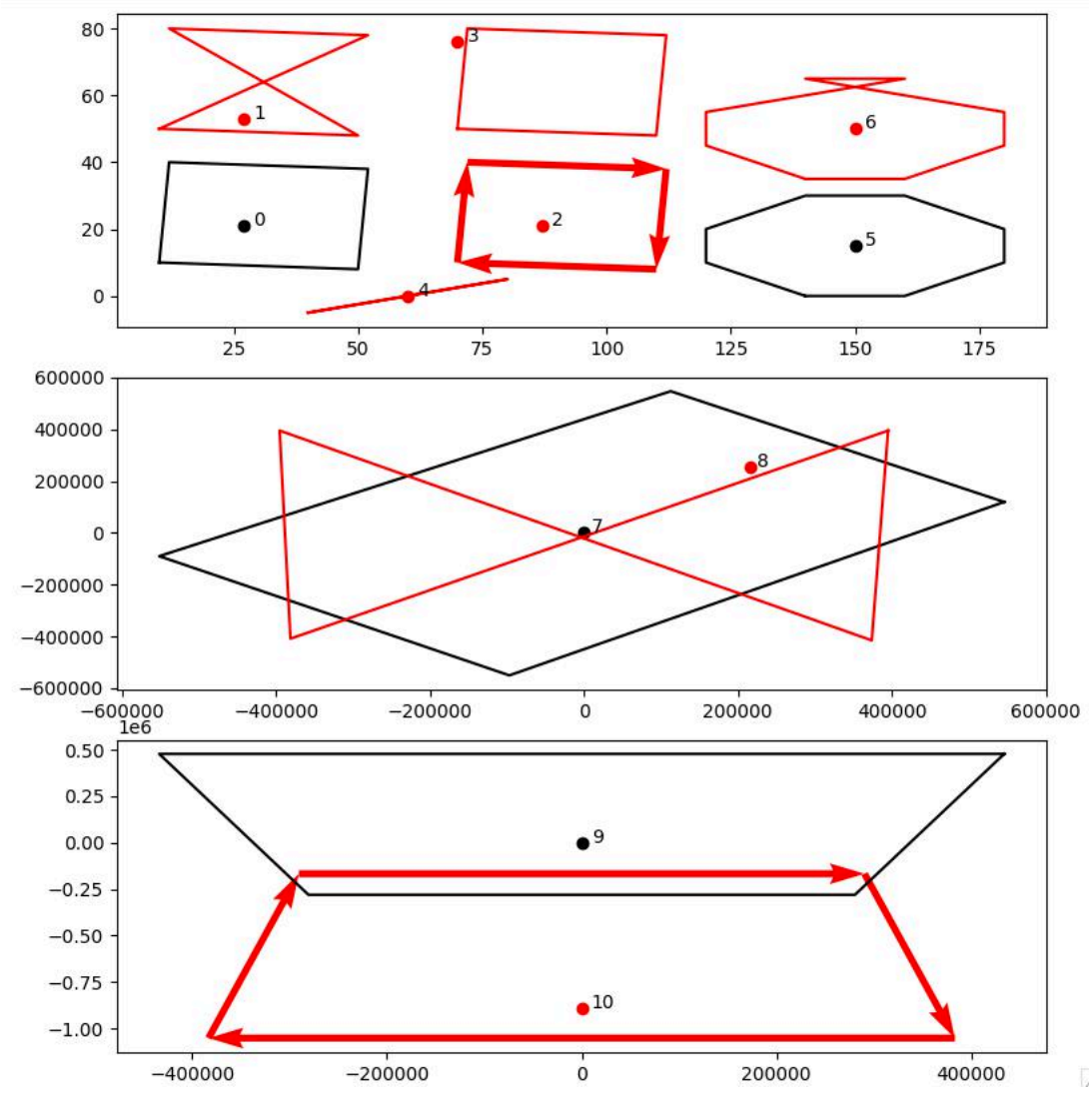
In order to overcome the aforementioned crash, a minimal set of XIOS initialization calls has been added to the `mpp_init` routine from `OPA_SRC/LBC/mppini.F90`. Notice that this

initialization is not complete yet, therefore the simulation will eventually crash in an XIOS call, but can at least reach the end of `oasis_enddef` and output the OASIS grid description files.

Grids checks

In order to check if the output grid description is coherent with the OASIS requirements, we use a python script relying on the `netCDF4` and `shapely` packages, `check_oasis_grid.py`. `Shapely` has the native capability of checking if a polygon is "simple" (sides are not crossing), if it is entered in counter-clockwise order and if it contains a point (its centre in our case). Since `shapely` works on projections, we use an azimuthal projection in the polar regions.

As shown in the figure, where the script is applied to a synthetic test case, the script works for any number of corners and is capable of detecting degenerated cells. The three pans represent the projected North Pole region, the mid latitudes and the projected South Pole region.



The analysis is also output on screen:

```
Problem on cell 1 (fortran nr. 2)
It is not convex. It is counterclockwise. It contains its centre.
coord [[10, 50], [50, 48], [12, 80], [52, 78]] centre [27, 53]

Problem on cell 2 (fortran nr. 3)
It is convex. It is not counterclockwise. It contains its centre.
coord [[72, 40], [112, 38], [110, 8], [70, 10]] centre [87, 21]

Problem on cell 3 (fortran nr. 4)
It is convex. It is counterclockwise. It does not contain its centre.
coord [[70, 50], [110, 48], [112, 78], [72, 80]] centre [70, 76]

Problem on cell 4 (fortran nr. 5)
It is not convex. It is counterclockwise. It contains its centre.
coord [[40, -5], [80, 5], [80, 5], [40, -5]] centre [60, 0]
```

The script, as it is, is not meant for public use and no effort has been made to drive it by arguments: the user input has to be provided by editing a few lines in the script

```
############################
# USER INPUT
############################

stop_on_error = False # Stop on first not compliant cell (useful dor dealing
with one error at a time)
first_cell =  0  # Restart cell (useful for dealing with one error at a time)
skip_mask = True # Do not check masked cells
oasis_files = True # Input from OASIS compliant files (naming conventions)

manual_input = False # Use handwritten values for testing the script itself

gin = 'nogt'
gmn = 'nogt'
if oasis_files:
    GridFile = './GRIDS_01_CHECK/grids.nc'
    MasksFile = './GRIDS_01_CHECK/masks.nc'
else:
    GridFile = './GRIDS_01_SINGLE_NEWLAT/grids_nemo_crn.nc'
    MasksFile = './GRIDS_01_SINGLE_NEWLAT/masks_nemo.nc'
```

Validation

The new routines `OPA_SRC/LBC/mppini.F90`, `OPA_SRC/DOM/domwri.F90` and `SBC/cpl_oasis3.F90` have been used to generate the grid description of the ORCA1 and ORCA025 configuration of the NEMO grids used in CNRM-CM6-1-LR and CNRM-CM6-1-HR, which use two different types of folding at the North. The resulting grid description has been validated by comparison with the grids used for the CMIP6 exercise (i.e. on belenos respectively in directories /scratch/climat/CMIP6/data/cpl/ and /scratch/climat/CMIP6/data/cpl/tl359_eORCA025L75).

The script `check_oasis_grid.py` was used to test all grids used in CNRM-CM6-1-LR and CNRM-CM6-1-HR and they all passed the test successfully. The new routines have been committed on the CNRM git repository on branch `v3_6_STABLE` and correspond to the git tag 54ba03a41b0bc3b91a588802ed61a56e7bb0d6c7 .

Conclusions

The routine cpl_oasis3.F90 used to produce the files containing a description of the position of the centre, corners, mask and area of each cell of the NEMO grid cells, needed by the OASIS coupler, has been revised. It can now be used in parallel or only on one process and provides good results for the different types and resolution of NEMO grids. A Python script has also been written to check the validity of the cells defined by the routine verifying if the cell is convex, if its corners are given counter-clockwise and if the centre of the cell is enclosed within its 4 corners. The routine has been used to generate the grid description of the ORCA1 and ORCA025 configuration of the NEMO grids used in CNRM-CM6, which use two different types of folding at the North, and the resulting grid description has been validated by comparison with the grids used in CMIP6 and with the Python script. It will now be used to generate the grid description of the ORCA12 grid.

References

- S. Valcke, T. Craig, E. Maisonnave, L. Coquart: *OASIS3-MCT User Guide, OASIS3-MCT 5.0*, CERFACS Technical Report TR/CMGC/21/161, 2021
- Gurvan Madec, and the NEMO team: *NEMO ocean engine book, version 3.6 stable*, Institut Pierre-Simon Laplace Technical Report No 27 ISSN No 1288-1619, 2016

Attached files

- `cpl_oasis3.F90`
- `domwri.F90`
- `mppini.F90`
- `rebuild_corners.py`
- `transform_oasis_grid_files_gen.ncl`
- `check_oasis_grid.py`