MOSAIX

Table of Content

MOSAIX is a small set of fortran routines, bash scripts, python scripts and xml files allowing to generate coupling weights for IPSL Earth System Model. It replaces the old MOSAIC software. It uses XIOS as a library to compute weights. In the present state, it can handle ORCA, LMDZ lon/lat and DYNAMICO grids.

MOSAIX can generate 1st and 2nd order conservative interpolation weights.

As MOSAIX uses XIOS, it works in parallel, using MPI distributed memory.

Compatibility with MOSAIC

- Weights are **not** fully compatible with the weights generated by MOSAIC. The precision of the interpolation (conservation) is improved. The ocean mask interpolated on the atmosphere grid is slighlty different.
- **Run-off** : runoff weights are computed with a different algorithm, the result can be quite different.
- **Calving** : In IPSLCM6, the calving was integrated for each latitude of the grid on the LMDZ grid. Weights were use to aggregate several latitudes to compute an integral over a latitudinal band.. This is not possible with DYNAMICO, and was abandon for rectilinear LMD grid. The atmosphere model must send the integral over a region (presently latitudinal bands). The final result should be the same as with MOSAIC.

TODO

- Creates a non-masked grid for atmosphere, to compute integrals of run-off
- Creates 2nd order interpolation weights that preserves posivity. They won't be conservative.

Future work

- Geographical distribution of the calving for the northern hemisphere.

Known problems

- ?

Validation

MOSAIX and the old method MOZAIC are very similar for temperature, albedo, ice cover and heat and water fluxes. However, MOSAIX generates weights that are slighlty different from those generated by MOZAIC. The global conservation is better.

For wind stress, the old method relies on OASIS bilinear interpolation (from the Scrip library). MOSAIX generates 2nd order weights.

For runoff, the method is different. We have run three simulations with IPSLCM6 LR to check the new weights :

- TEST.CM6.MOSAIX.11 : old MOZAIC weights
- TEST.CM6.MOSAIX.18 : new MOSAIC weigths with runoff parameters `atmCoastWidth=2 oceCoastWidth=3 searchRadius=600.0`. Weight files are `IGCM/CPL/IPSLCM6/eORCA1.2xLMD144142/rmp_*MOSAIX_v0.nc`
- TEST.CM6.MOSAIX.19 : new MOSAIC weigths with runoff parameters `atmCoastWidth=2 oceCoastWidth=2 searchRadius=550.0`. Weight files are `IGCM/CPL/IPSLCM6/eORCA1.2xLMD144142/rmp_*MOSAIX_v1.nc`

Results available at `tgcc:$CCCSTOREDIR/../../gencmip6/p86caub/IGCM_OUT/IPSLCM6/DEVT/piControl`

An intermonitoring is available at ■http://webservices2017.ipsl.fr/interMonitoring/tmp/interMonitoring_plot01_pAPyBD_prod/

A CESMEP atlas is under preparation.

Code availability

- Web : ■https://forge.ipsl.jussieu.fr/igcmg/browser/TOOLS/MOSAIX
- SVN : `svn co http://forge.ipsl.jussieu.fr/igcmg/svn/TOOLS/MOSAIX MOSAIX`
- SVN : `svn co svn+ssh://<login>@forge.ipsl.jussieu.fr/ipsl/forge/projets/igcmg/svn/TOOLS/MOSAIX MOSAIX`

Portability

MOSAIX has been tested on Curie (old TGCC computer) and Irene (■TGCC), and on a Mac.

Requirements

- Fortran compiler, C++ compiler, MPI library.
- A working ■XIOS library. Revision 1615 is known to work.
- fcm version 1.2
- NetCDF fortran librairies. See ■XIOS for a suitable version.
- Python, version>=3.6 with ■netCDF4 and ■numpy modules. Python version>=2.7 mights work, but has not been tested recently.
- Bash version>=4.
- ■NCO. Version 4.6.0 is known to work.

Input files

MOSAIX needs files describing the grid of the models. See `CreateWeightsMask.bash` and the bottom of this page.

Output files

All netCDF files have a detailed header.

- Weight files `rmp_*`. Should have a rather explicit name. The name is configurable at the beginning of `CreateWeightsMask.bash`.
- Diagnostic file used for debug `dia_*.nc` for each rmp file, except for runoff and calving. Debug information are directly in weight files in this last case.
- Grid description file for OASIS-MCT `grids.nc`, `areas.nc` and `masks.nc`.
- A file containing the fraction of ocean in each atmosphere grid box. For instance `ICO40_grid_maskFrom_eORCA1.2.nc`.
- A `README` file. Includes a checksum for each generated files.

Scripts and codes

- **CreateWeightsMask.bash**. Creates the weights to interpolate between atmospheric and ocean grid. Weight files are suitable for OASIS-MCT. Creates the fraction of ocean in atmospheric grid cells. Uses parallel processing with MPI. Has a configuration section at the beginning.
- **CreateOasisGrids.bash**. Creates files `grids.nc`, `areas.nc` and `masks.nc` with information from both models, suitable for OASIS-MCT. Mono CPU. Launched by `CreateWeights.bash`.
- **update_xml.py**. Python script used to perform on the fly editing of xml files. Used by `CreateWeights.bash`. More information with `python update_xml.py -h`.
- **iodef_atm_to_oce.xml** and **iodef_oce_to_atm.xml**. xml files read by `interpol.f90`. These files are edited by `update_xml.py`.
- **MOSAIX/src/MOSAIX/interpol.f90**. Fortran source using the XIOS library.

  make_mosaix. Small script for compiling **interpol.f90**. You need to compile XIOS first. Uses compiler, compiler options and librairies defined by XIOS.

```
make_xios [options]
options :
      [--xios path] : path to XIOS. Default is ../XIOS
      [--full]      : to regenerate dependencies and recompile from scratch
Example : ./make_mosaix --xios ../XIOS
```

- **RunoffWeights.py**. Python code to generates weights for run-off. Launched by `CreateWeights.bash`. More information with `python RunOffWeights.py -h`.

  1) Defines a coastal band on land in the atmosphere model. For LMDZ rectilinear grids, the width of this band is parametrized, in grid points. For ico grids, the band contains only point with a fraction of ocean in]0,1[.

  2) Defines a coastal band on ocean in the ocean model. The width of this band is parametrized.

  3) An atmosphere point of the coastal band is linked to an ocean point in the coastal band if the distance between the two is less than `searchRadius`.

- **CalvingWeights.py**. Python code to generates weights for calving. Launched by `CreateWeights.bash`. More information with `python CalvingWeights.py -h`.

1) The atmosphere model integrate the calving over on specific regions. Presently the regions are three latitudinal bands with limits 90°S/40°S/50°N/90°N.

2) The weights distribute the calving in the ocean in the corresponding latitudinal bands. By default, the distribution is uniform.

3) For the southernmost band, it is possible to use a geographical distribution read in a file. These files are currently available for eORCA1 and eORCA025.

Grids

- *[tuv]orc*. ORCA grid at *[TUV]* points. Masked on land, with area equal to grid box surface.
- *tlmd*. LMD rectilinear grid at scalar (physics) point. Masked on land, with area equal to grid box surface.
- *tico*. LMD icosahedron grid at scalar (physics) point. Masked on land, with area equal to grid box surface.
- *olmd*. LMD rectilinear grid at scalar (physics) point. Not masked, with areas set to 1.
- *oico*. LMD icosahedron grid at scalar (physics) point. Not masked, with areas set to 1.
- *corc*. ORCA grid at *T* point. Masked on land, duplicated (from periodicity) point masked, with area equal to grid box surface.
- *oorc*. ORCA grid at *T* point. Masked on land, duplicated (from periodicity) point masked, with area equal 1.

Generating input grid files

Generating icosahedral grid in double precision using XIOS:

Here are the main changes:

- File `context_lmdz.xml` :

```
.....
<domain_definition>
    <domain id="dom_glo" data_dim="2" />
    <domain id="dom_glo_p8"  domain_ref="dom_glo"  prec="8" />
    <domain id="greordered"  domain_ref="dom_glo">
    <reorder_domain invert_lat="true" shift_lon_fraction="0"/>
   </domain>
</domain_definition>

...

<grid_definition>
.....
    <grid id="grid_glo">
       <domain domain_ref="dom_glo" />
    </grid>

    <grid id="grid_glo_p8">
       <domain domain_ref="dom_glo_p8" />
    </grid>
.....
```

- File `field_def_lmdz.xml` :

```
<field_definition level="1" prec="4" operation="average" freq_op="1ts" enabled=".true." default_value="9.96921e+36" domain
   <field_group id="fields_2D_p8" grid_ref="grid_glo_p8" prec="8">
     <field id="aire_p8" field_ref="aire" long_name="Grid area p8" unit="-"    />
     <field id="fract_oce_p8" field_ref="fract_oce" long_name="Fraction oce p8"   unit="1" />
     <field id="fract_sic_p8" field_ref="fract_sic"   long_name="Fraction sic p8"   unit="1" />
     <field id="fract_oce_plus_sic_p8" long_name="Fraction oce plus sic p8"   unit="1" />
     <field id="mask_oce_plus_sic_p8" long_name="Masque oce plus sic p8"  unit="1" />
   </field_group>
[...]
```

```
</field_definition>
```

- File `file_def_histmth_lmdz.xml` :

```
<file_definition>
   <file_group id="defile">
       <file id="grid_p8" name="grid_p8" output_freq="1d" sync_freq="10d" output_level="2" enabled="true" timeseries="none
         <field_group grid_ref="grid_glo_p8" ts_enabled="true" freq_op="1d" expr="@this">
             <field field_ref="aire_p8" level="1" operation="once"  />
             <field field_ref="fract_oce_p8" level="1" operation="once" />
             <field field_ref="fract_sic_p8" level="1" operation="once" />
         </field_group>

         <field_group grid_ref="grid_glo_p8" ts_enabled="true" freq_op="1d" >
             <field field_ref="fract_oce_plus_sic_p8" level="1" > fract_oce + fract_sic </field>
             <field field_ref="mask_oce_plus_sic_p8" level="1" > ((fract_oce + fract_sic) &gt; 0) ? 1 : 0 </field>
         </field_group>
       </file>
.........
```