tutorial_communication,

a tutorial on how to instrument a code with OASIS3-MCT API calls

September 2021

Introduction

This directory contains the files of a tutorial to learn how to instrument a code with calls to the OASIS3-MCT library so to couple it with other components. The tutorial involves two toy model codes, `ocean.F90` and `atmos.F90`, into which you are invited to implement the calls to OASIS3-MCT API (Application Program Interface) routines. Toy models are skeleton programs that do not contain any real physics or dynamics but that can reproduce real exchanges of coupling fields. Instrumenting those toy models gives a practical experience of using the OASIS3-MCT library.

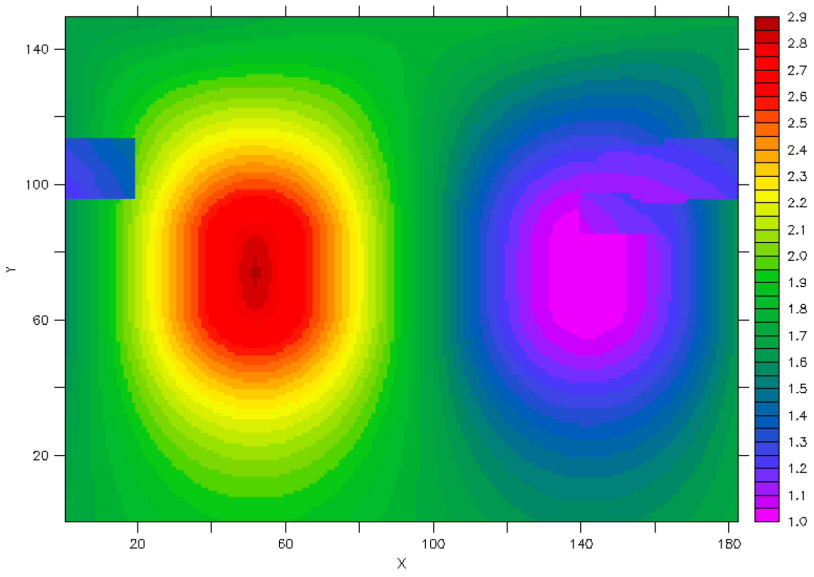
This tutorial is extracted from the Short Private Online Course (SPOC) on “Code Coupling with OASIS3-MCT” developed in the framework of the ESiWACE Centre of Excellence. This SPOC is composed of videos, quizzes and hands-on. If you are interested in attending the SPOC, please visit the online training section of CERFACS web site at https://cerfacs.fr/online-training/. Videos and quizzes were also extracted from the SPOC and are available as Open Education Resources (OER) material at https://www.oercommons.org/courseware/lesson/85340 .

OASIS3-MCT is a coupling library that can be used to exchange information, the so-called coupling fields, between different codes or even within one same code. Internally, OASIS3-MCT uses the Model Coupling Toolkit (MCT) developed by the Argonne National Laboratory. Usually, the codes to be coupled represent different parts of the Earth System, e.g. the ocean or the atmosphere. The codes can be sequential, i.e. run on only one process, or parallel, i.e. run on more than one process. Each code must be compiled and linked with the OASIS3-MCT library. During the simulation, the OASIS3-MCT library performs the exchange of coupling fields between the different executables (i.e. the compiled codes). When the code is parallel, each process can send or receive its part of the coupling field (i.e. there is no need to gather the whole coupling field on one process). When the OASIS3-MCT library performs the exchanges, it can apply different transformations on the coupling fields, for example spatial regridding (i.e transforming the coupling field from the grid of the source code to the grid of the target code).

The toy coupled model to implement

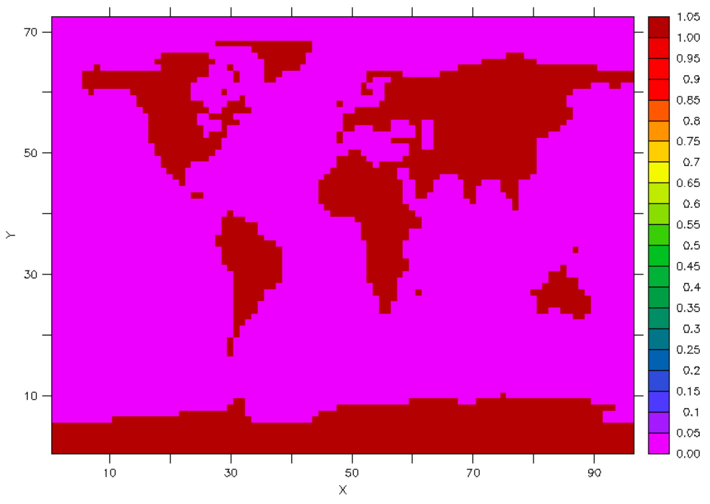
The objective of the tutorial is to instrument the codes of the toy models provided (`ocean.F90` and `atmos.F90`) to implement basic coupling exchanges between those toy models. In their original form, `ocean` and `atmos` run for 4 hours (14400 seconds) without exchanging any coupling field; `ocean` has a timestep of 3600 sec and `atmos` a timestep of 1800 sec.

The grid of `ocean` has 182x149 grid points globally but `ocean` is parallel so each ocean process defines its local partition and reads only its part of the grid in the file `ocean_mesh.nc`. Then each process allocates two local arrays `field_send_ocean`, a coupling field to send and `field_recv_ocean` the coupling field to receive. The coupling field `field_send_ocean` is function of the longitude and the latitude of the grid points and of the timestep number, so it varies at each timestep. At the first timestep, `field_send_ocean` looks like this (in the 182x149 grid point space):



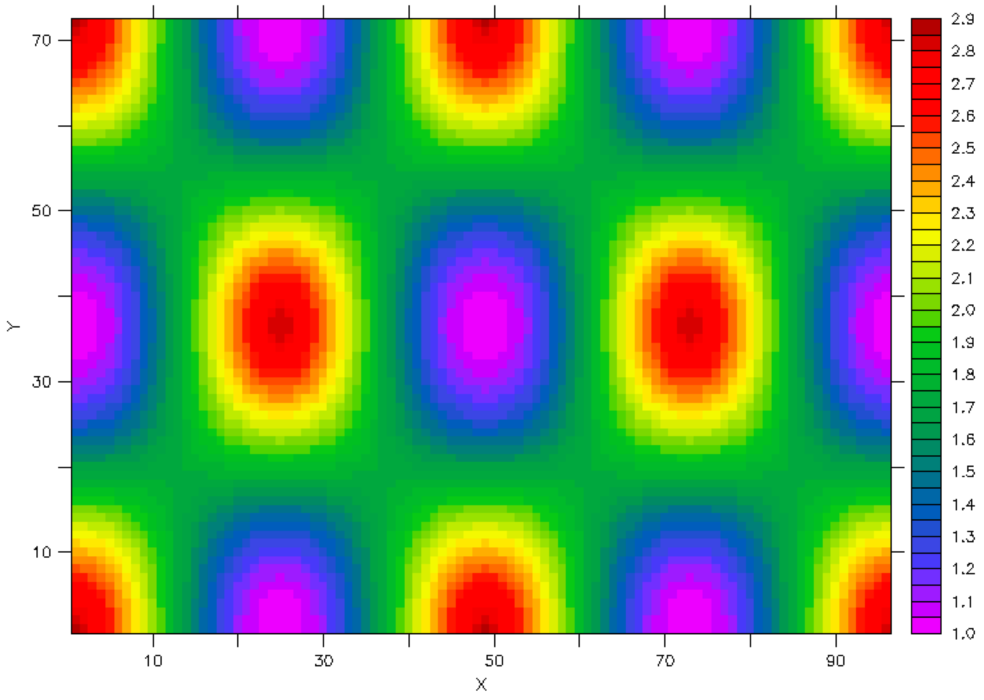
Note that the square-like patterns are due to the distortion of the grid over the continents.

The grid of `atmos` has 96 x 72 grid points globally but, as `ocean`, `atmos` is parallel so each `atmos` process defines its local partition and reads only its part of the grid in the file `atmos_mesh.nc`. The next figure illustrates the sea-land mask on the `atmos` grid. Following OASIS3-MCTconvention, masked cells have a value of 1 (red) and non-masked cells a value of 0.

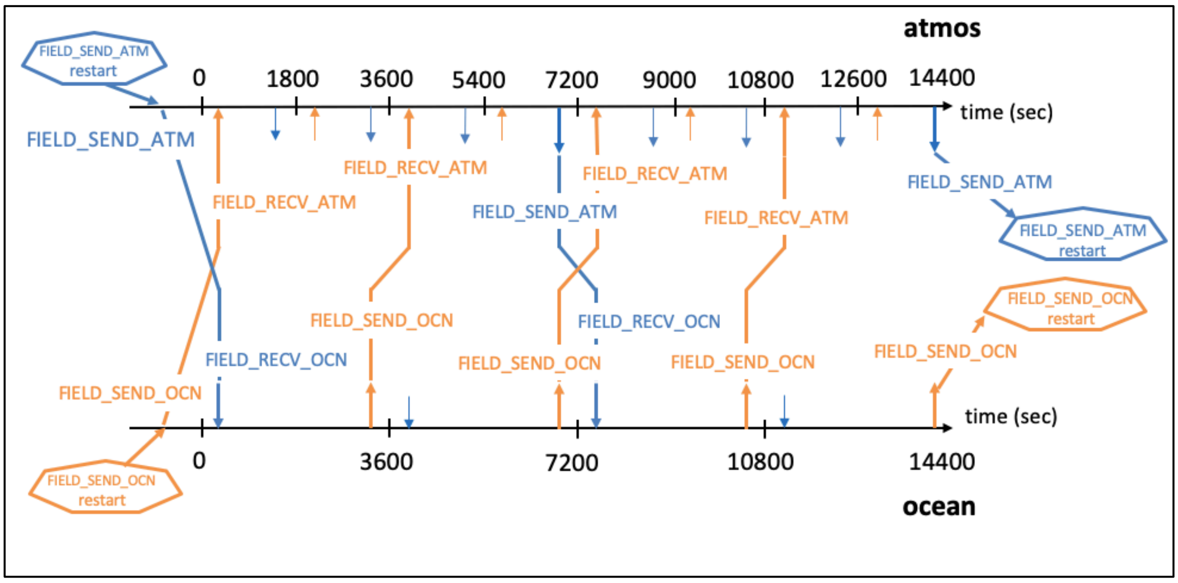


LMDZ.MSK

Then each process allocates two local arrays `field_send_atmos`, the coupling field to send and `field_recv_atmos` the coupling field to receive. The coupling field `field_send_atmos` is defined with a function that varies with the longitude and the latitude faster than in ocean and it also varies in time. At the first timestep, `field_sent_atmos` looks like this (in the 96 x 72 grid point space):



The coupling exchanges to implement are illustrated here. The `ocean` sends its array `field_send_ocean` (with symbolic name `FIELD_SEND_OCEAN`) to `atmos` which receives it in the array `field_recv_atmos` (with symbolic name `FIELD_RECV_ATMOS`); the coupling period is 3600 sec. In return, `atmos` sends its array `field_send_atmos` (with symbolic name `FIELD_SEND_ATMOS`), to `ocean` which receives it in the array `field_recv_ocean` (with symbolic name `FIELD_RECV_OCEAN`); the coupling period is 1800 sec.



<u>Compiling and running “empty” tutorial toy models</u>

To compile the toy model components, you first have to compile OASIS3-MCT on your computing platform (see section 6 of the User Guide). Then to compile the toy model components, simply type, in the tutorial_communication directory:

```
      > make clean; make
```

Note that the Makefile in this directory automatically includes your OASIS3-MCT header makefile – see the first line in the Makefile. The executables `atmos` and `ocean` should be available in the current directory.

To run the two tutorial components as separate models, edit the script `run_tutorial` to adapt it to your platform and execute it

```
      >./run_tutorial
```

The results of the component models should now be in subdirectory `$rundir` defined in `run_tutorial`. In their current form, `atmos.F90` and `ocean.F90` are not interfaced with the OASIS3-MCT library and do not perform any exchanges. They just run, on 4 processes each for 4 model hours without doing anything specific. Have a look at the files `atmos.out_10?` and `ocean.out_10?` (where `?` goes from 0 to 3, one for each process), in the `$rundir` subdirectory.

<u>Interfacing the toy models with OASIS3-MCT and running it</u>

The tutorial toy model should reproduce the coupling exchanges between `atmos` and `ocean` as described above. To do so, modify `atmos.F90` and `ocean.F90` so to implement the initialisation, definition and declaration calls of OASIS3-MCT library. In the time step of the models, implement the reception of the input coupling fields (`oasis_get`) at the beginning of the time step and the sending of the output coupling field (`oasis_put`) at the end of the time step. The lines where to introduce the OASIS3-MCT specific calls are marked with a line specifying the call to implement, for example:

```
!!!!!!!!!!!!!!OASIS_DEF_PARTITION!!!!!!!!!!!!!!!!!
```

Once instrumented, recompile `atmos.F90` and `ocean.F90` . To run the two tutorial components as a coupled models, edit the script `run_tutorial_oa` to adapt it to your platform and execute it

```
      >./run_tutorial_oa
```

Note that `run_tutorial_oa` copies the configuration file `namcouple`_LAG from directory `/data_tutorial`, in the `$rundir` directory as `namcouple`. This `namcouple` specifies that a coupling field with symbolic name `FIELD_SEND_OCN` has to be matched with a coupling field with symbolic name `FIELD_RECV_ATM` and that the mapping between the two fields has to use the pre-existing weight and addresses file `my_remapping_file_bilinear.nc`. It also specifies that a coupling field with symbolic name `FIELD_SEND_ATM` has to be matched with a coupling field with symbolic name `FIELD_RECV_OCN` and that the interpolation to transform the field has to be the `SCRIPR/BILINEAR` one (i.e. the weight and addresses file used for the interpolation will be calculated in the initialisation phase with the SCRIP library).

The results of the tutorial coupled model should now be in your `$rundir` subdirectory. The file `nout.000000`, written by the master process of one model, contains the information read in the configuration file `namcouple`. The OASIS3-MCT debug files `debug.01.00000?` and `debug.02.00000?` (where `?` goes from 0 to 3, one for each process) are written by each process of `ocean` and `atmos` respectively. The level of debug information in these files corresponds to the value of the first number on the line below `$NLOGPRT` in the `namcouple`. If your run was successful, these debug files should all end with :

```
(oasis_terminate) SUCCESSFUL RUN
 -- EXIT  (oasis_terminate)
```

We hope that you have successfully instrumented the tutorial toy models and run them in coupled mode. If you have not, please note that a version of the toy model codes instrumented with OASIS3-MCT API calls is available in the directory, see `atmos.F90_oa` and `ocean.F90_oa.`

If you need further help, do not hesitate to consult the OER material available at https://www.oercommons.org/courseware/lesson/85340 and/or to follow the SPOC on “Code Coupling with OASIS3-MCT” (see https://cerfacs.fr/online-training/). We are also available at oasishelp@cerfacs.fr to answer questions you may have.

Have fun with OASIS3-MCT !