OASIS3-MCT training

Cerfacs - 18-19/10/2018

All the documentation about the coupler OASIS3-MCT can be found on the OASIS web site at http://oasis.enes.org and in the OASIS3-MCT sources in the oasis3-mct/doc directory.
The current OASIS3-MCT coupler uses internally the Model Coupling Toolkit (MCT) developed by the Argonne National Laboratory (http ://www.mcs.anl.gov/mct) to perform parallel regridding and parallel exchanges of the coupling fields.

A. Extracting OASIS3-MCT sources

To login on the training room computers, choose Other and enter your login and password (e.g. form01 and form01). Each participant must login with a different login and password.
On internet, go to the OASIS web site to register and download the sources at:
https://verc.enes.org/oasis/download/oasis-registration-form
Open a terminal and use the command svn checkout you get from the web site :

- cd $HOME
- mkdir oasis3-mct
- cd oasis3-mct

Execute the svn command

B. Compiling OASIS3-MCT and running an empty "tutorial" toy model

1. To compile the OASIS3-MCT coupler:
   - Load the appropriate modules by editing your .bashrc adding: “module load intel intelmpi” and by executing it with : « source .bashrc »
   - Go into directory oasis3-mct/util/make_dir
   - Adapt "make.inc" to include your platform header makefile , i.e. make.intelmpi18_training for the training room computers
   - Adapt the value of $COUPLE and $ARCHDIR in your platform header makefile
   - Compile OASIS3-MCT with "make –f TopMakefileOasis3" (to recompile from scratch use "make realclean –f TopMakefileOasis3").
   - The libraries "libmct.a", "libmpeu.a", "libpsmile.MPI1.a" and "libscrip.a" that need to be linked to the models are available in the directory $ARCHDIR/lib
2. To compile the tutorial models:
   - Go into directory oasis3-mct/examples/tutorial
   - Type "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 model1 and model2 are available in the current directory.
3. To run the two tutorial component models:
   - Edit the script run_tutorial to adapt it to your platform “training_computer” and execute it
     > ./run_tutorial
     The results of the component models are now in subdirectory $rundir defined in run_tutorial.

- ➢ In their current form, "model1.F90" and "model2.F90" are not interfaced with the OASIS3-MCT library and do not perform any exchanges. They just run for 6 time steps of 1 hour each without doing anything specific (see the files "model1.out_100" and "model2.out_100" in the output directory work_tutorial)
- Rename your “work_tutorial” in “work_tutorial_notcoupled” to save it for comparison later.

C. <u>Interfacing the "tutorial" toy model with OASIS3-MCT and running it</u>

To interface the model1 and model2 codes, you must understand the coupling algorithm that this toy should reproduce.

The "tutorial" toy model should reproduce the coupling between two simple codes, model1 and model2, with the OASIS3-MCT coupler.

The total run is 6 hours (21600 seconds). Model1 has a time step of 3600 seconds and runs on the logically-rectangular ORCA2 grid (182x149). Model2 has a time step of 1800 seconds and runs on the logically-rectangular LMDz grid (96x72).

The coupling exchanges to implement are as follows: a field with symbolic name “FSENDOCN” in model1 and “FRECVATM” in model2 is sent from model1 to model2 every 2 hours and a field with symbolic name “FSENDATM” in model2 and “FRECVOCN” in model1 is sent from model2 to model1 every 3 hours:

- At the beginning of the run, model1 receives “FRECVOCN” coming from a restart file; at the end of a coupling period of 2 hours, model1 sends the coupling field “FSENDOCN” to model2 that receives it as “FRECVATM” at the beginning of the next coupling period (see figure fsendocn below).
- In addition, model1 also outputs to a file the time averaged field "FSENDOCN" every 2 hours (see the figure fsendocn_file).
- At the beginning of the run, model2 receives “FRECVATM” coming from a restart file; at the end of the coupling period of 3 hours, model2 sends the coupling field “FSENDATM” to model1 that receives it as “FRECVOCN” at the beginning of the next coupling period (see the figure fsendatm below).
- The interpolation from "FSENDOCN" into "FRECVATM" uses a pre-existing weight and addresses file called "my_remapping_file_bilinear.nc". The interpolation specified to transform "FSENDATM" into "FRECVOCN" is the SCRIPR/BILINEAR one (i.e. the weight and addresses file used for the interpolation are calculated at the first coupling time step with the SCRIP library).

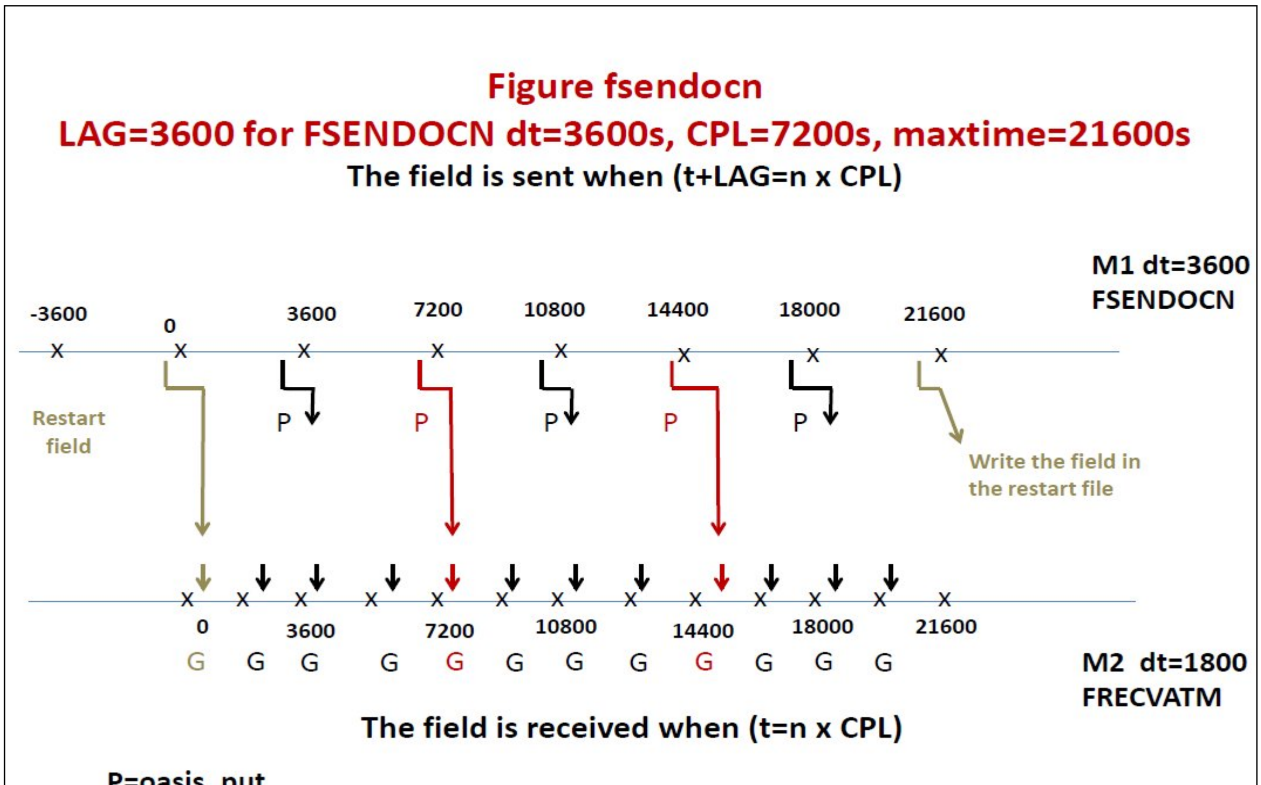


Figure fsendocn
LAG=3600 for FSENDOCN dt=3600s, CPL=7200s, maxtime=21600s
The field is sent when (t+LAG=n x CPL)
M1 dt=3600
FSENDOCN
-3600
0
3600
7200
10800
14400
18000
21600
Restart field
P
Write the field in the restart file
0
3600
7200
10800
14400
18000
21600
G
M2 dt=1800
FRECVATM
The field is received when (t=n x CPL)
P=oasis_put

Figure fsendocn_file

FSENDOCN sent to a file CPL=7200s, maxtime=21600s

The field is sent when (t=n x CPL)

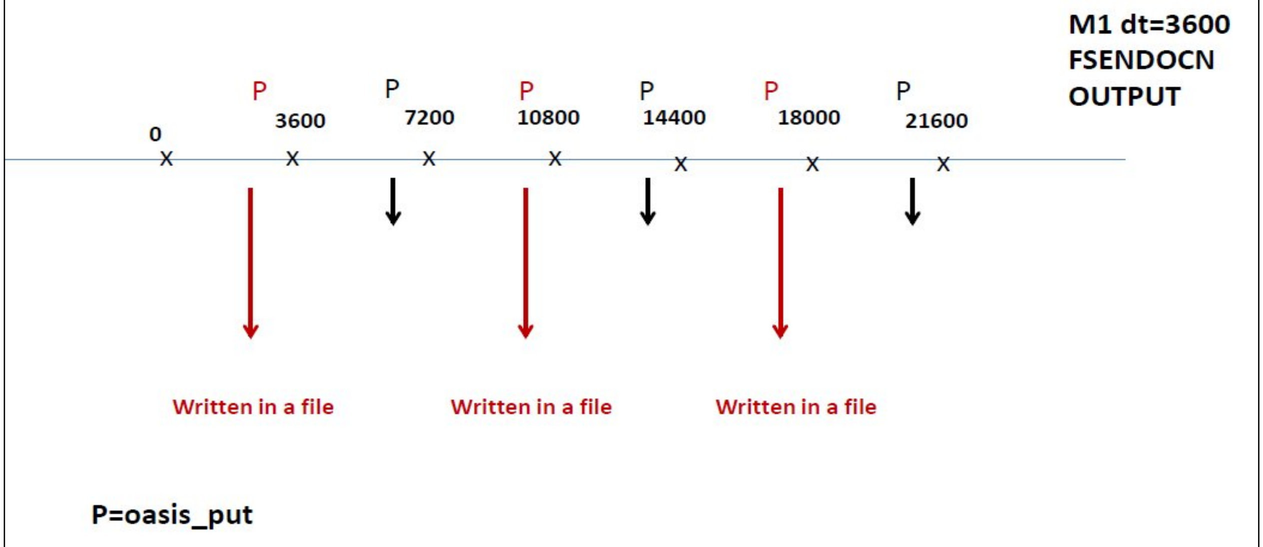
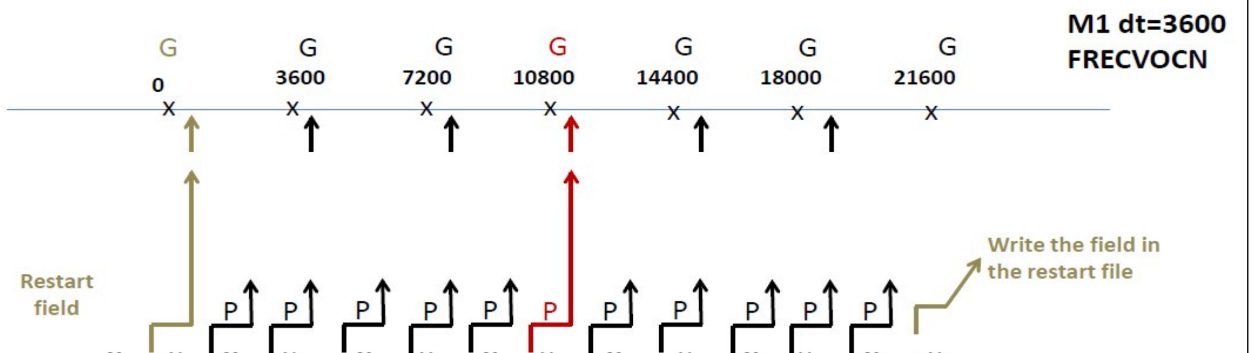


Figure fsendatm

LAG=1800 for FSENDATM dt=1800s, CPL=10800s, maxtime=21600s

The field is received when (t=n x CPL)



Modify "model1.F90" and "model2.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 as described above (see figures fsendocn and fsendatm). The lines where to introduce the OASIS3-MCT specific calls are marked with ""!! TOCOMPLETE ...". As a first step, suppose that each model will run on only one process. Most of the variables that will be used in the oasis routines calls are already defined at the beginning of the programs.

Recompile model1.F90" and "model2.F90" as described above.

Have a look at the OASIS3-MCT configuration file "*namcouple*", located in the directory oasis3-mct/examples/tutorial/data_oasis3, written to perform the exchanges described above. The interpolation specified to transform "FSENDATM" into "FRECVOCN" is 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). Interpolation from "FSENDOCN" into "FRECVATM" will use a pre-existing weight and addresses file (my_remapping_file_bilinear.nc).

Run the toy coupled model with the script "run_tutorial".

- The results of the tutorial coupled model are now in your work 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.root.01 and debug.root.02 are written by the master process of each model. The level of debug information in these files corresponds to the value of the first number on the line below $NLOGPRT in the *namcouple* (see the section 3.2 of the User Guide for more details). If the run is successful, you should get "SUCCESSFUL RUN" at the end of these debug files.

- You can visualize the content of the netCDF debug files (produced as the coupling fields have the "EXPOUT" status in the *namcouple*) with ferret and the scripts script_ferret_*.jnl.

- Explain why the number of time steps in the file FSENDATM_model2_03.nc is not the same than in the file FRECVOCN_model1_03.nc.

- Keep your results by renaming your working directory

In this configuration, model1 and model2 run with one process each. This is indicated in the launching script "run_tutorial" (see "nproc_exe1" and "nproc_exe2").

To run and couple model1 and model2 in parallel, one has to ensure that the partition definition transferred to OASIS3-MCT via the call to oasis_def_partition is properly done. In model1 and model2, the information to provide as arguments of the oasis_def_partition is calculated by the subroutine decomp_def.F90 that you can check in more detail. Then, the number of processes has to be modified in the launching procedure.

- Specify for example "nproc_exe1=3", "nproc_exe2=3" in the script run_tutorial .

- In oasis3-mct/examples/tutorial, execute "run_tutorial", which will then launch the models on 3 processes each.
- ➢ Keep your results by renaming your working directory
- ➢ Visually compare the results with the non-parallel case

OASIS3-MCT supports different parallel decompositions for the models (see the section 2.4 of the User Guide). Tutorial routine "decomp_def.F90" can define the local partition for the BOX or the APPLE decompositions. By default, the APPLE decomposition is used. To test the BOX decomposition, recompile (make clean ; make) the tutorial models with, in Makefile: "CPPKEYDECOMP_M1=DECOMP_BOX" or "CPPKEYDECOMP_M2=DECOMP_BOX".

D. Using "test_interpolation" to calculate interpolation weights and checking their quality

The test_interpolation environment offers a practical example on how to use OASIS3-MCT_4.0 to pre-calculate (i.e. in a separate job prior to the "real" simulation) the remapping weight&address file. Test_interpolation couples two component models, model1 and model2, and evaluates the quality of the remapping between the source grid of model1 and the target grid of model2 by calculating the remapping error on the target grid. Only one coupling exchange is performed at t=0 when model1 sends its coupling field FSENDANA, which is received as FRECVANA by model2. The field values of FSENDANA are defined by an analytical function on model1 grid (torc). The remapping error is defined as the difference between the interpolated values of FRECVANA and the values of the analytical function on the target grid (lmdz) points, divided by the interpolated field (and multiplied by 100 to have it in %). As specified in the configuration file data_oasis3/namcouple, OASIS3-MCT performs a conservative remapping FRACNNEI between torc and lmdz grids. These grids are defined in the OASIS3-MCT grid data files "grids.nc", "masks.nc" and "areas.nc", in the subdirectory /data_oasis3 .

To compile test_interpolation models:

- Go into directory oasis3-mct/examples/test_interpolation
- Type "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 model1 and model2 are available in the current directory.

Then adapt the script run_testinterp.sh to your platform. For the first test, keep the SRC_GRID, TGT_GRID and remap as defined (respectively torc, lmdz, conserv). Of course, to run on another architecture then the ones already included (nemo Lenovo, kraken, training_computer at Cerfacs and Météo-France beaufix), you would have to adapt all parts of the script, which are architecture specific.

In OASIS3-MCT_4.0, a hybrid MPI/OpenMP parallel version of the SCRIP remapping library is available with the coupler sources. To calculate the remapping weights and addresses in parallel, OASIS3-MCT_4.0 relies on the MPI parallel layout of the calling model but only enrols one MPI process per node. The number of OpenMP threads per node is set by a dedicated environment variable OASIS_OMP_NUM_THREADS; for optimum performance and to avoid hyper threading, it is recommended to set this variable to the number of cores of the node. Notice that, for most of the OpenMP implementations, the number of threads activated at run time is limited by the overall value set by the OMP_NUM_THREADS environment variable. If OASIS_OMP_NUM_THREADS is not set, it defaults to OMP_NUM_THREADS.

All details about the hybrid MPI/OpenMP parallel version of the SCRIP can be found in Piacentini et al. 2018.[1]

You can now run the script, with "./run_testinterp.sh 1_2_1" (i.e. 1 node, 2 MPI tasks –one for each model, 1 OpenMP thread per MPI task) or activating the OpenMP threading e.g. with "./run_testinterp.sh 1_2_8" (i.e. 1 node, 2 MPI tasks –one for each model, 8 OpenMP thread per MPI task)

Results are in the $rundir directory as defined in the script. If the run is successful, the following files are present: the remapping file rmp_torc_to_lmdz_CONSERV_FRACNNEI.nc, the analytical field on the source grid torc file FSENDANA_model1_01.nc, the interpolated field on the target grid lmdz in file FRECVANA_model2_01.nc, and the error field as defined above in file error_interp.nc. You can find the minimum and maximum of the error and other diagnostics in the output file model2.out_101.

E. Test_interpolation for other couple of grids and other remappings

To adapt test_interpolation to your own grids and remappings, you have to go through the following steps

- Generate the OASIS3-MCT grid data files (i.e. grids.nc, masks.nc and areas.nc) with the definition of your own grids, see the User Guide section 5.1 for details. One can adapt sources in test_interpolation/create_grids_masks_with_F90 and test_interpolation/create_grids_masks_with_NCL to create grids.nc, masks.nc and areas.nc (based on "data/grid_model1.nc" and "data/grid_model2.nc"), with either with F90 or NCL.
- Adapt the namcouple configuration file in /data_oasis3 directory for your own grids and remapping, see User Guide Chapter 3 for details.
- Change the SRC_GRID, TGT_GRID, and remap values in the script test_interp.sh

You should then be able to run test_interpolation for your grids and remapping and evaluate the quality of the remapping by analysing the error field in error_interp.nc.

[1] https://cerfacs.fr/wp-content/uploads/2018/03/GLOBC-TR-PIANCENTINI-cmgc_18_34.pdf

Defining what a “good” remapping precisely means is impossible. But at least, the test_interpolation environment should allow you to identify bugs (if any) and to compare the quality of different remappings.

Of course, you can also ask the OASIS developers for advice regarding your specific results!