OASIS3-MCT tutorial

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

Go to the OASIS web site to register and download the sources at:
https://portal.enes.org/oasis/download/download-oasis3-mct-sources
Use the git clone command from the web site :

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

Execute the git command

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

1. To compile the OASIS3-MCT coupler:
   - Go into directory `oasis3-mct/util/make_dir`
   - Adapt the "`make.inc`" file to include your platform header makefile
   - Adapt the value of $COUPLE and $ARCHDIR in your platform header makefile
   - Load appropriate modules, if needed
   - Compile OASIS3-MCT by typing
     ```
     > make realclean -f  TopMakefileOasis3
     ```
     and then
     ```
     > make -f  TopMakefileOasis3
     ```
   - The libraries "`libmct.a`", "`libmpeu.a`", "`libpsmile.MPI1.a`" and "`libscrip.a`" that need to be linked to the models are available in the directory `$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.
   - To run the two tutorial component models:
   - Edit the script `run_tutorial` to adapt it to your platform 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 subdirectory `work_tutorial`)

C. Interfacing the "tutorial" toy model with OASIS3-MCT and running it

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 12 hours (43200 seconds). Model1 has a time step of 3600 seconds and runs on the logically-rectangular ORCA2T grid (182x149). Model2 has a time step of 1800 seconds and runs on the logically-rectangular LMDz grid (96x72).

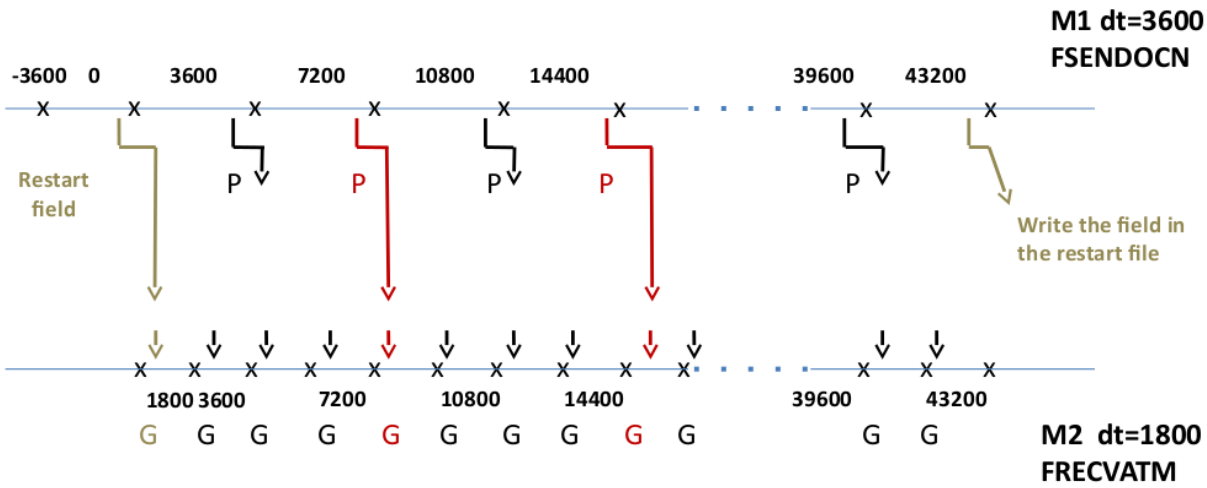
The coupling exchanges to implement are as follows: a field with **symbolic name** “FSENDOCN” in model1 and “FRECVATM” in model2 is sent from model1 to model2 every 2 hours and a field with **symbolic name** “FSENDATM” in model2 and “FRECVOCN” in model1 is sent from model2 to model1 every 3 hours:

- At the beginning of the run, model1 receives “FRECVOCN” coming from a restart file; at the end of the 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=43200s

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



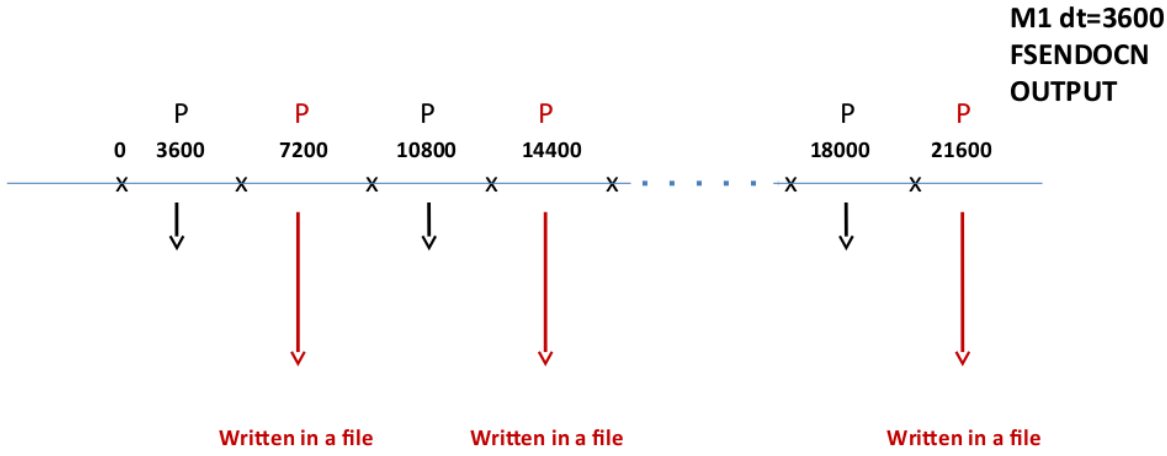
The field is received when (t=n x CPL)

P=oasis_put
G=oasis_get

Figure fsendocn_file

FSENDOCN sent to a file CPL=7200s, maxtime=43200s

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

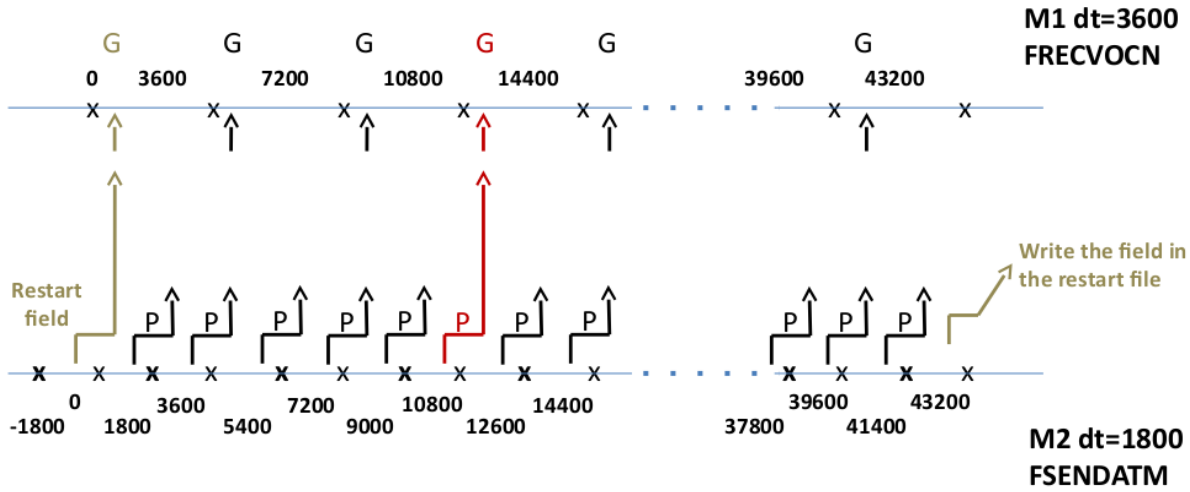


P=oasis_put

Figure fsendatm

LAG=1800 for FSENDATM dt=1800s, CPL=10800s, maxtime=43200s

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



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

P=oasis_put
G=oasis_get

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).
- You can visualize the content of the netCDF debug files (produced as the coupling fields have the "EXPOUT" status in the namcouple) with the FERRET visualisation toolkit, calling the programs written in FERRET language 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_tutoria .
- In oasis3-mct/examples/tutorial, execute "run_tutorial", which then launches the models on 3 processes each.
- Keep your results by renaming your working directory
- Visually compare the results with the non-parallel case

Keep your results by renaming your working directory

D) Load imbalance

Coupled systems usually run slower that the stand alone models which compose it. They are several reasons for that. In the case of concurrent running of the components (like in this tutorial), the main reason is the load imbalance: one of the two components (model1 or model2) is running slower than the other, that has to wait, at each coupling time

step. This also generate a waste of resources (the computing cores of the waiting component are switched on for nothing). In the following exercise, we propose to evaluate and reduce this load imbalance.

A post processing tool, named LUCIA, is provided with the OASIS suite. During the simulation, tracing files are produced by the library, in the working directory. In a post-processing phase, the LUCIA tool diagnoses, for each component of the coupled system, the time spent waiting for coupling fields. This time depends on the speed differences between the components. The load imbalance will be reduced to a minimum when all components will run at the same speed.

- In the `oasis3-mct/util/lucia` directory, compile the LUCIA tool launching:
  ```
  >./lucia -c
  ```

- In the `oasis3-mct/examples/tutorial/data_oasis3/namcouple` file, add a second value equal to -1 to the $LOGPRT section (see § 3.2 “First section of namcouple file” of the User Guide). This enable the LUCIA tracing

- In the `oasis3-mct/examples/tutorial` directory, launch the coupled system:
  ```
  > ./run_tutorial
  ```

- Check that the LUCIA tracing files were produced in the working directory. In this directory, execute the LUCIA post-processing tool:
  ```
  > $HOME/oasis3-mct/util/lucia/lucia
  ```

- Evaluate the load imbalance thanks to the standard output of LUCIA or visualising graphically the computing and waiting time of model1 and model2:
  ```
  > display oasis_balance.eps
  ```

- Find the best couple of decomposition (`nproc_exe1,nproc_exe2`) to minimize the waiting times