A collaboration between
Environment Canada MSC/RPN and CERFACS

Coupling MEC with IML ocean model through OASIS3-GOSSIP2

Sophie Valcke

November, 14th 2005
CERFACS Technical Report
TR/CMGC/05/111

Contents

Chapter 1

Acknowledgements

I would like to thank all people at RPN for the warm and friendly atmosphere at work, especially Gilbert Brunet, director of RPN at the time the collaboration started in 2004, for his constant efforts and support to realize this collaboration, and Pierre Pellerin, current RPN director, for his warm welcome and personal implication during the visit.

Chapter 2

Introduction

Since July 2004, a collaboration is established between the Numerical Prediction Research Division (RPN) of Meteorological Service of Canada (MSC) and the European Centre for Research and Advanced Training in Scientific Computation (CERFACS). The goal of the collaboration is to merge European and Canadian efforts in the development of coupling software used for environmental modelling. This collaboration will optimize the efforts and should naturally increase the scientific exchanges between the two groups.

RPN develops since 1997 a coupling software called the "RPN-coupler", using in particular the GOSSIP communication layer based on Unix sockets. The RPN-coupler was used to couple atmospheric models GEM (Global Environmental Model) and MC2 (Mesoscale Compressible Community) to different environmental models (ocean, sea ice, wave, hydrology, etc.). Independently, CERFACS develops, since more then 10 years now, the OASIS coupler that is currently used by about 20 climate modelling groups in Europe but also in the USA, in Australia, Japan, India, and China. OASIS forms the core of the European project PRISM which objective is to provide a standard software infrastructure for climate modelling in Europe (1).

A detailed comparison of the two couplers realized in 2003 concluded that they were indeed providing the same type of services, i.e. communication and regridding, for the same type of applications (2). In fact, both RPN-coupler and OASIS allow synchronized exchanges of information between numerical models and provide the transformations needed to express, on the grid on the target model, the information provided by the source model on its grid.

The collaboration started by a one-year visit of Sophie Valcke from CERFACS to RPN from September 2004 until August 2005. The objectives of the visit were to evaluate the feasibility of merging European and Canadian efforts in the coupler development and to start the fusion. During this period, Sophie Valcke spent 60% of her time on the collaboration. The work realized during this period is described in this report.

As a first step in the collaboration, it was decided to interface RPN socket based communication layer, GOSSIP2, in the latest version of OASIS3; this work is described in chapter 3. OASIS3 was first ported and tested with a toy coupled model on RPN platforms (see 3.1). Then, the technical work of introducing GOSSIP2 in OASIS3 took place, with help from Djamel Bouhemhem, resulting in the merged OASIS3-GOSSIP2 coupler (see 3.2). Some time was also spent, mainly with Serge Desjardins, on defining a standard maintenance and compiling procedure of OASIS3-GOSSIP2 at RPN (see 3.3).

The second part of the visit was devoted to the set-up of a Coupled General Circulation Model (CGCM) based on the atmospheric component GEM of the Modelling Environmental Community (MEC) System and on the Global Ocean Model (GOM) from Institut Maurice Lamontagne. Active interaction with Pierre Pellerin, Manon Faucher and Serge Desjardins took place during this period. To set-up the coupled model, it was necessary to interface the OASIS3-GOSSIP2 model interface library in the component models. The work realized for MEC is presented in chapter 4: the modifications in the dynamics and in the physics are described respectively in 4.1 and in 4.2, details on the adaptation of MEC namelist are given in 4.3, and

the testing procedure of those modifications coupling two MECs together is described in 4.4.

The interfacing of the GOM model and the set-up of the MEC-GOM coupled model per se were done mainly by Manon Faucher and are not described in detail in this report. However, the general architecture of the MEC-GOM coupled model is described in chapter 5 with a detailed description of the coupling algorithm (see 5.1) and of the coupling fields exchanged between the atmosphere and ocean models (see 5.2). Some issues in the MEC-GOM coupling deserving particular attention are also detailed in 5.3.

Finally, some conclusions and a perspective for the future of the collaboration are given in 6.

Chapter 3

OASIS3-GOSSIP2 at RPN

During the first part of the visit, OASIS3 was ported on RPN platforms and tested with its toy coupled model (see 3.1). Then the RPN socket based communication layer, GOSSIP2, was interfaced in OASIS3 (see 3.2). Some details on the OASIS3-GOSSIP2 compiling procedure at RPN are also given in 3.3.

At CERFACS, OASIS3 was mainly developed in the framework of the European project PRISM and is currently maintained under the PRISM Support Initiative. PRISM objective is to provide a standard software infrastructure for climate modelling in Europe. OASIS3, PRISM and GOSSIP2 are not described in detail in this report. For more information on OASIS3, see (5); on OASIS3 toy coupled model, see (6); on PRISM Standard Compiling Environment (SCE), see (3); on PRISM Standard Running Environment (SRE), see (4); on GOSSIP2, see (7).

3.1 Porting OASIS3 and its toy coupled model on RPN platforms

At run-time, OASIS3 acts as a separate mono process binary which drives the coupled run, interpolates and transforms the coupling fields; we refered to this binary as the OASIS3 *interpolator binary*. To communicate with OASIS3, a component model needs to be linked with the OASIS3*PSMILe library*. The first part of the work was to port and test the OASIS3 interpolator binary and a toy coupled model (in which of course the component models use the PSMILe library) on some RPN platforms.

3.1.1 Steps

The following steps were necessary to port and test OASIS3 and its toy coupled model on Linux and SGI platform at RPN:

- Adaptation of PRISM Standard Compiling Environment (SCE) for Linux platform *armc28*: creation of sub-directory `include_armc28` in PRISM directory tree `prism/util/compile/frames/`
- Adaptation of PRISM SCE for SGI platform *pollux*: creation of sub-directory `include_pollux` in `prism/util/compile/frames/`
- Adaptation of PRISM SCE for Linux platform *rossby*: creation of sub-directory `include_rossby` in `prism/util/compile/frames/`
- Personal installation of NetCDF library for Linux and for SGI
- Personal installation of `ncview` on Linux to visualize NetCDF files (`~armngr1/userlibs/ncview/ncview-1.92e/ncview`)
- Installation of tkcvs, graphical interface to the source manager CVS, used by PRISM and CERFACS to manage OASIS3 sources
- Adaptation of PRISM Standard Running Environment (SRE) for Linux platform *armc28*: creation of sub-directory `include_armc28` in `prism/util/running/frames/`

- Adaptation of PRISM SRE for SGI platform *pollux*: creation of sub-directory `include_pollux` in `prism/util/running/frames/`

3.1.2 Issues

The following issues still require some attention:

- The SCE was adapted to allow compilation of OASIS3 interpolator binary and PSMILe library with 4-byte REAL precision, except for the PSMILe IO library `mpp_io` for which it did not work out. No time was spent to look into this problem and all tests in single precision did not use the PSMILe IO functionality (compilation with cpp key "key_noIO").
- Library mpich-1.2.4 was used on Linux. Error first occured with message: "undefined reference to '__ctype_b"'. M. Valin suggested to link with C-ctype.o (currently in `/users/dor/armn/gr1 /userlibs/Linux/comp_mpich`. This is currently included in SCE for Linux platform *armc28*.
- An official NetCDF library should be installed at RPN.

3.1.3 Results

In conclusion, the following results were obtained:

- OASIS3 toy coupled model was compiled (8-byte REAL precision) and successfully run on Linux platform *armc28* with MPI1 communication technique (mpich-1.2.4) (parallel models communicating in parallel with OASIS3 interpolator binary).
- OASIS3 toy coupled model was compiled (8-byte REAL precision) and successfully run on SGI *pollux* with MPI1 and MPI2 communication technique (lam mpi) (parallel models communicating in parallel with OASIS3 main process).
- Compiling environments for Linux platforms *armc28* and *rossby*, and SGI *pollux* were checked in the official OASIS3 CVS repository at CERFACS.

3.2 Interfacing GOSSIP2 in OASIS3 and other OASIS3 adaptations

As a first step in the merging of OASIS3 and the RPN coupler, it was decided to introduce the RPN socket based communication layer, GOSSIP2, in OASIS3. GOSSIP2 can therefore be used to do the communication between the the PSMILe library and the OASIS3 interpolator binary; this is decided at compile time and is totally transparent for the model itself.

3.2.1 Steps

The following steps were realized:

- GOSSIP2 standard test was compiled and run.
- OASIS3 `clim` library, normally using Message Passing Interface (MPI) for all communication, was adapted to use GOSSIP2 interface routines (MGI); a new `clim.GSIP` library was checked in OASIS3 CVS in `prism/src/ lib/clim.GSIP` .
- OASIS3 PSMILe library was adapted to use MGI (still keeping the MPI option) for all communication: modified sources were checked in OASIS3 CVS in `prism/src/lib/psmile`.
- GOSSIP2 and MGI sources were checked in OASIS3 CVS in `prism/src/lib/gossip`.
- A field exchanged between two models travels from the source model to the GOSSIP2 Server, then from the Server to OASIS3 interpolator binary where it is transformed, then from OASIS3 binary to the Server and finally from the Server to the target model. Ideally, the GOSSIP2 Server could be merged into the OASIS3 interpolator binary. This would reduce the amount of communication, but

would not be trivial to realize. As the performance of the communication was not considered crucial, it was decided to keep the GOSSIP2 Server as is. The GOSSIP2 Server sources were checked in OASIS3 CVS in `prism/util/running/toygossip/Server`.

- SCE was adaptated for *armc28* and *rossby* Linux platforms to compile OASIS3 and the toy models with PSMILe using GOSSIP2 (use <mp>='GSIP' when generating OASIS3 and toymodel compilation scripts with `prism/util/compile/frames/Create_COMP_Models.sh`).
- GOSSIP2 sources, compilation script (using RPN environment) and running scripts were checked in `prism/util/running/toygossip/Server`; coherence with sources in `prism/src/lib/gossip` is automatically ensured by links.
- A problem was detected when testing OASIS3 toymodel running with 3, 3, 1, and 1 processes for respectively toyatm, toyche, toyoce and OASIS3: the maximum number of threads used by the Server was too small; the problem was corrected by D. Bouhemhem (02/11/2004).
- A problem was detected in the transfer of 8-byte REAL variables; the problem was corrected by D. Bouhemhem (14/12/2004).
- A new PSMILe routine `prism_get_freq` was written to allow a component model to obtain the coupling frequency of a particular coupling field; this functionality was needed in MEC for its coupling algorithm.
- The PSMILe routines that can be used by the model to provide its grid information to OASIS3 interpolation binary (`prism_write_grid`, `prism_write_corner`, `prism_write_mask`, `prism_write_area`) were modified. Previously, all the component models were, one at a time, accessing the auxiliary files to write their grid information (`grids.nc, masks.nc, areas.nc`). Currently, with the GSIP communication technique, the component models send their grid information to OASIS3 interpolation binary which then writes it to the appropriate auxiliary files. This improvement is for now not included for the MPI1 or MPI2 communication technique.

3.2.2 Issues

It is important to be aware of the following issues:

- With GSIP communication technique, direct communication between the models (without going through OASIS3 interpolator binary) is not possible.
- For communication between OASIS3 and the models, only one channel per model process, and not one channel per coupling field, is used. This limits the number of channels used but implies that **all coupling fields have to be send and received in the order into which they are listed in the OASIS3 configuring file** `namcouple`.
- On Linux platforms, using OASIS3 GSIP to couple parallel models does not work if all model processes communicate with OASIS3 main process. M. Valin suspects a problem of spurious interaction between the sockets used by GOSSIP2 and the sockets used by the MPI library (mpich-1.2.4 implementation) for the model internal communication. This problem was not solved.
- Tests on Linux however showed that using OASIS3 GSIP to couple parallel models does work if only the master model process communicates with OASIS3 interpolator binary. (In that case however, "1 1" must be specified below "GSIP" in the `namcouple` in the `$CHANNEL` section.)
- Porting of OASIS3-GOSSIP2 on the IBM at RPN is currently under work; note that the coupler has already run on the IBM Power4 at the European Centre for Medium Range Weather Forecast.

3.2.3 Results

- OASIS3 toy coupled model was compiled (8-byte or 4-byte REAL precision) and successfully run on Linux platform *armc28* with GSIP communication technique (mono-process toy component models only, or parallel toy component models with only the master process communicating).

- GOSSIP2 adaptation was checked in official OASIS3 CVS repository at CERFACS (refered to as the “GSIP communication technique”)

3.3 Compiling OASIS3-GOSSIP2 at RPN

For technical reasons, it was decided to use PRISM SCE to compile OASIS3-GOSSIP2 at RPN instead of adapting it to RPN compiling environment (*ÉTAGÈRE*); this will also facilitate the upgrading with future OASIS3 versions delivered by CERFACS.

3.3.1 Steps

The following steps were realized to define a standard compiling procedure at RPN for OASIS3-GOSSIP2.

- S. Desjardins extracted OASIS3-GOSSIP2 source code from CVS server at CERFACS into `$ARMNLIB/modeles/SURF/oasis/v_3.2.5/prism`. This directory will be refered to as `~prism` in this document.
- To compile OASIS3 interpolator binary, one has simply to follow steps described in OASIS3 documentation; see (5).
- A new script, `scr_compile` was created into a new directory `~prism/src/mod/MECoasis` (this directory is not present in prism standard tree) to compile OASIS3 model interface library PSMILe. It compiles PSMILe sources from directory `~prism/ src/lib/psmile`, GOSSIP2 and MGI sources (`gossip_sock.c` and `mgilib2.c` including `gossip.h` and `mgi.h`) from directory `~prism/ src/lib/gossip` and the MEC-PSMILe interface routines (`c_oasis_init .ftn90`, `c_oasis _partition.ftn90, c_oasis_put.ftn90, c_oasis_get.ftn90` written for MEC to encapsulate PSMILe F90 modules, see also 4.1) also from the new directory `~prism/ src/mod/MECoasis`. The resulting library, that must be linked to the component models is `~prism/src/mod/MECoasis/ malibLinux/libpsmileRPN.a`.

3.3.2 Issues

On the longer term, one should take care of the following points:

- To ensure coherency between the PSMILe library and the GOSSIP2 server, the script `scr_compile` should be modified to automatically use the GOSSIP2 sources `gossip_sock.c`, `gossip.h`, and `mgi.h` used by the GOSSIP2 Server used at RPN. Currently, the most up-to-date version of the Server at RPN is in `~armnbou/Public/Coupler/Server`.
- The MEC-PSMILe interface routines and the PSMILe routines called directly in MEC code (i.e. `prism_def_var_proto`, `prism_enddef_proto`, and `prism_terminate_proto`) were included in GEM 3.2.1 as stub routines. They should be removed in the next version and linking with the `~prism/src/mod/MECoasis/ malibLinux/libpsmileRPN.a` should be automatic.

3.3.3 Results

The official versions of OASIS3-GOSSIP2 interpolator binary and PSMILe library at RPN are respectively:

- `~prism/$ARCH/bin/ oasis3.GSIP.x`
- `~prism/src/mod/MECoasis/malib$ARCH/ libpsmileRPN.a`

Chapter 4

Coupling modifications in MEC

This chapter describes the modifications realized in MEC to set-up the coupled model based on MEC and GOM component models.

Some of these modifications are needed to extract appropriate information from the physics or to feedback coupling information to the physics[1]. These modifications must of course be activated to couple MEC with other models via the OASIS3-GOSSIP2 coupler[2], but can also be used to include other modules in MEC code, such as sea-ice or hydrological modules[3].

The final version of the modified MEC sources were given to Manon Faucher and Serge Desjardins. An intermediate version of the modifications was also included in the GEM 3.2.1 version (July 2005). The phasing between the final version and GEM 3.2.1 still needs to be done (see also 3.3.2 and 5.3).

4.1 Modifications in the dynamics

Figure 4.1 shows the directory tree of modified or new MEC routines in the dynamics. The MEC-PSMILe interface routines (which names start by `c_oasis_`) are also shown on the figure as well as the PSMILe routines (which names start by `prism_`); these routines are not part of MEC sources, but are present in directories `~prism/src/mod/MECoasis` and `~prism/ src/lib/psmile` respectively (see section 3.3.1).

The description of the modifications will follow the same tree.

The modified comdecks are listed in figure 4.2.

4.1.1 Subroutines `set_world_view1.ftn` and `bcastc1.ftn`

Subroutine `set_world_view1.ftn` was sligthly modified to include new coupling namelist initialisation (see also 4.3). In `bcastc1.ftn`, some broadcasts of variables not used anymore (`C_pipe, C_nml, C_step`) were removed.

4.1.2 Subroutines `c_init_cplfld.ftn`: coupling field definition

The subroutine `c_init_cplfld.ftn`, called by `p_init.ftn`, defines the coupling fields and performs some general initialization.

The logic followed in `c_init_cplfld.ftn` is illustrated in figure 4.3.

[1] These modifications are set by the key `C_coupling_L=.true.` in the `&coupling` part of MEC namelist (see section 4.3).

[2] In this case, the key `C_cploasis_L` must also be `.true.` in MEC namelist.(see section 4.3)

[3] These modules should then be directly called in routine `c_cplg_surf.ftn` (see section 4.1.9).

```
gemdm.ftn
  |-->set_world_view.ftn
       |--> set_world_view1.ftn
       |--> bcastc1.ftn
  |--> set_cn1.ftn
        |--> p_init.ftn
               |--> c_init_cplfld.ftn
                      |--> c_finddim.ftn
               |--> p_config.ftn
                      |--> phyopt.ftn
                      |--> inikey.ftn
  |--> gem_ctrl.ftn
         |--> indata.ftn
                |--> readgeo.ftn
                |--> rdrstrt.ftn
         |--> c_set_coupling.ftn
                |--> c_oasis_init.ftn90
                       |--> prism_init_comp_proto
                       |--> prism_start_grids_writing
                       |--> prism_write_grid
                       |--> prism_write_mask
                       |--> prism_terminate_grids_writing
                |--> c_init_oasisfld.ftn
                       |--> c_oasis_partition.ftn90
                              |--> prism_def_partition_proto
                       |--> prism_def_var_proto
                       |--> prism_enddef_proto
         |--> gem_run.ftn
                |--> hdif_phy.ftn
                       |--> p_main.ftn
                              |--> p_phystep.ftn
                                     |--> p_physlb.ftn
                                            |--> c_fillbus.ftn
                                            |--> c_getbus.ftn
                              |--> c_cplg_surf.ftn
                                     |--> c_oasis_put.ftn90
                                            |--> prism_put_proto
                                     |--> c_mtlmap.ftn
                                     |--> c_oasis_get.ftn90
                                            |--> prism_get_proto
                                     |--> c_mtlmap.ftn
                |--> wrrstrt.ftn
  |--> c_stop_coupling.ftn
         |--> prism_terminate_proto
```

Figure 4.1: Tree of modified routines in MEC dynamic code

```
busind.cdk
c_cplg.cdk
nml.cdk
nmlcpl.cdk
sfcbus.cdk
```

Figure 4.2: Modified comdecks in MEC dynamic code

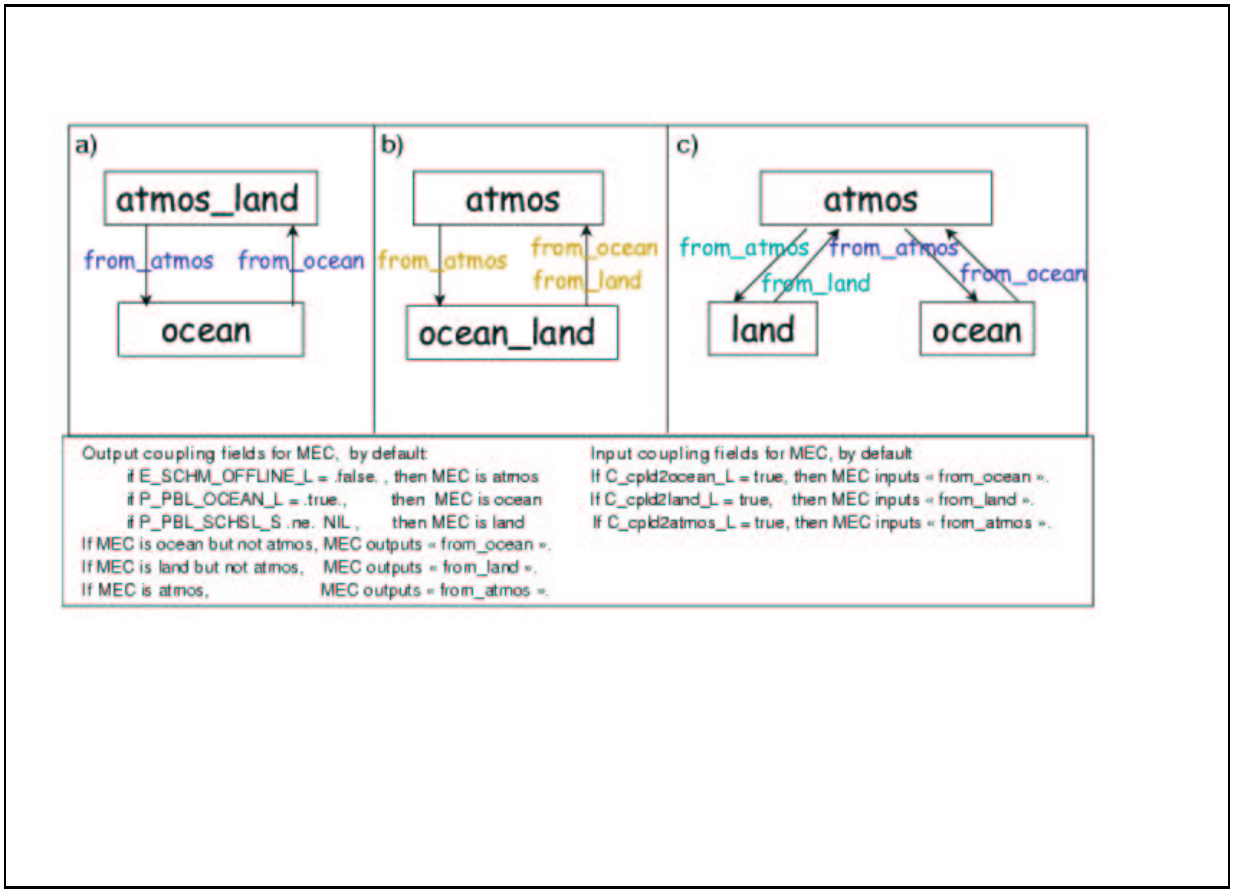

Figure 4.3: Input and output coupling field definition in MEC

Sets of fields: “from_atmos”, “from_ocean”, and “from_land”

Three sets of fields are defined, “from_atmos”, “from_ocean”, and “from_land”. In MEC, these sets are defined by default[4] as follows:

- Set “from_atmos”
  - TJA: screen level temperature
  - UDA: screen level X-component of wind
  - VDA: screen level Y-component of wind
  - T8A: temperature at t-dt
  - U8A: wind speed along-x at t-dt
  - V8A: wind speed along-Y at t-dt
  - TDA: screen level specific humidity
  - P8A: surface pressure at t-dt
  - H8A: specific humidity at t-dt
  - FBA: solar energy flux towards ground (shortwave)

[4]These sets can be redefined by the user in MEC namelist (see section 4.3)

  - FIA: infrared energy flux towards ground (longwave)
  - RTA: total precipitation rate

- Set “from_ocean”
  - GLW: sea ice fraction
  - I8W: sea ice thickness
  - I7W: sea ice temperature
  - TMW: sea surface temperature
  - FCW: surface sensible heat flux over water
  - FVW: surface latent heat flux over water
  - MCW: coupling mask over water
  - FCI: surface sensible heat flux over ice
  - FVI: surface latent heat flux over ice
  - MCI: coupling mask over ice
- Set “from_land”
  - FCL: surface sensible heat flux over land
  - FVL: surface latent heat flux over land
  - MCL: coupling mask over land
  - FCG: surface sensible heat flux over glaciers
  - FVG: surface latent heat flux over glaciers
  - MCG: coupling mask over glaciers

MEC coupling role and corresponding output fields definition

MEC can “play” different coupling “roles”. For example, MEC can run its atmospheric dynamic and physic cores, but MEC can also, in “offline” mode, run only its ocean module and/or a land surface scheme. The definition of the output coupling fields (sent by MEC) depends on its “coupling role”, whereas the definition of the input coupling fields (received by MEC) depends on the models to which MEC is coupled (see next paragraph). MEC coupling roles (“atmos”, “ocean”, and/or “land”) is defined by default in `c_init_cplfld.ftn` by the following analysis:

- If logical `schm_offline_L` is false, MEC runs its atmospheric parts and therefore plays an “atmos” role, and
- if logical `P_pbl_ocean_L` is true, MEC runs its ocean module and therefore plays an “ocean” role, and
- if variable `P_pbl_schsl_S` is not equal to `NIL`, MEC runs a land surface scheme and therefore plays a land role.

Note that MEC coupling roles can be redefined by the user by setting `lg_atmos` or `lg_ocean` or `lg_land` to `.true.` in MEC namelist (see section 4.3).

MEC coupling role defines its output coupling fields in the following manner:

- If MEC plays an “atmos” role, MEC outputs “from_atmos” fields, and
- if MEC plays an “ocean” role but not an “atmos” role, MEC outputs “from_ocean” fields, and
- if MEC plays an “land” role but not an “atmos” role, MEC outputs “from_land” fields.

Note that MEC can cumulate more than one coupling role and therefore can output more than one set of coupling fields.

Figure 4.3 a) corresponds to the MEC-GOM coupling; in this case, MEC plays “atmos” and “land” roles and therefore outputs “from_atmos” fields.

Input fields definition

The definition of the input coupling fields (received by MEC) depends on the models to which MEC is coupled; this must be explicitely defined by the user in MEC namelist:

- If MEC is coupled to an ocean model, `C_cpld2ocean⊥` should be true, and MEC is expected to receive "from_ocean" fields, and
- if MEC is coupled to a land model, `C_cpld2land⊥` should be true, and MEC is expected to receive "from_land" fields, and
- if MEC is coupled to an atmos model, `C_cpld2atmos⊥` should be true, and MEC is expected to receive "from_atmos" fields.

For the MEC-GOM coupling, MEC is coupled to an ocean model; therefore `C_cpld2ocean⊥` is true while `C_cpld2land⊥` and `C_cpld2atmos⊥` are false, and MEC receives "from_ocean" fields.

More details about `c_init_cplfld.ftn`

The following actions are achieved in `c_init_cplfld.ftn`:

- MEC role is defined by default based on values of logicals `schm_offline⊥` and `P_pbl_ocean⊥` and variable `P_pbl_schsl_S` (see above).
- The `&coupling` part of the MEC namelist is read and information is broadcast to all processes.
- If coupling is activated (`C_coupling⊥=.true`), general initialization is performed.
- If coupling is activated, the coupling fields in each set of fields exchanged (`from_atmos` and/or `from_ocean` and/or `from_land`) are defined: if no fields are specified by the user in the namelist, default fields are defined (see above).
- Index and position of each output/input field in the arrays containing all output/input fields, `rga_fldout`/`rga_fldin`, are defined.
- An array of integers, `iga_cplfld(6,4)`, is initialized. The element of this array equals 1 if the corresponding field is a input coupling field in the current simulation. The first dimension of this array refers to the following fields:
  1. surface sensible heat flux (FC)
  2. surface latent heat flux (FV)
  3. momentum flux coefficient (BM)
  4. albedo (AL)
  5. radiative temperature (TG)
  6. coupling mask (MC)

  whereas the second dimension refers to the type of surface:
  1. land(L)
  2. glaciers (G)
  3. water (W)
  4. ice (I)

  For example, `iga_cplfld(2,3)=1` means that the surface latent heat flux over water is an input coupling field in the current simulation. This array of integers is transfered to the physics in `p_phystep.ftn` where it is used to determine whether or not the corresponding flux can be overwritten by the coupling field (see 4.2.5).
- The arrays `sfc_out_S`/`sfc_in_S` containing the names of all output/input fields for the current simulation is created.
- The arrays containing all output/input fields, `rga_fldout` /`rga_fldin` are allocated.

Issues in `c_init_cplfld.ftn`

- In the set of fields "from_ocean", the fluxes and the mask are defined both over the sea water (FCW, FVW, MCW) and over the sea ice (FCI, FVI, MCI). Ideally, they should be split into two sets of fields "from_water" and "from_ice" to allow a coupling with two different models, an ocean model sending water fluxes and a sea ice model sending sea ice fluxes. In the current setting MEC-GOM, this is not required as GOM is both an ocean model and a sea ice (see also section 5.2).
- Similarly, the set of fields "from_land" should ideally be split into a set "from_land" for fields over land (FCL, FVL and MCL) and "from_glaciers" for fields over glaciers (FCG, FVG, and MCG). This is again not required for the current MEC-GOM coupling as "from_land" is not exchanged.
- To couple a MEC playing the "atmos" role with a MEC playing the "ocean" and "land" role (figure 4.3 b), the momentum flux coefficient, the albedo, and the radiative temperature over water, ice, land, and glaciers (BMW, BMI, BML, BMG, ALW, ALI, ALL, ALG, TGW, TGI, TGL, TGG) should be exchanged but these fields are currently not fully supported as coupling fields in MEC.

4.1.3 Subroutine `p_config.ftn`

This subroutines allows the transfer of information from/to the physics. It was modified to:

- give the value of the logical `C_coupling_L` to logical `COUPLING` in the physics;
- retrieve the values of the indices of surface types (soil, glacier, water, ice, aggregated value) in physics.

4.1.4 Subroutine `inikey.ftn`

This subroutine was modified to retrieve the indices of the following fields in the following buses:

- `fccpl, fvcpl, mccpl` added (in coupling case) in the dynamic bus `F_busdyn` (see also 4.2.4),
- `tdew` in the volatile bus `F_busvol`,
- `tdiag, udiag, vdiag, qdiag` in the permanent bus `F_busper`,
- `tmice, pr, rt, fc,fv` in the volatile bus `F_busvol`.

4.1.5 Subroutine `readgeo.ftn`: user-defined coupling mask MCPL

This routine was modified to allow the user to define explicitely a coupling mask as a geophysical fields. The user-defined coupling mask must be named MCPL[5]. If MCPL is present in the geophysical fields, it is read in and fills in the real array `rga_msk`; if not, `rga_msk` is filled in with the sea-land mask MGEN.

The array `rga_msk` is later used in `c_set_coupling.ftn` to define the coupling mask for OASIS (see 4.1.7) and in `c_cplg_surf.ftn` to define the coupling mask sent as an output coupling field ("MCA" in the MEC-GOM coupling, see 4.1.9). It is however currently not used to identified the region that must be overwritten by the flux coupling fields in the physics; this later mask comes from the mask input coupling fields ("MCW" and "MCI" in the MEC-GOM coupling, see 4.1.9 and also 5.3).

4.1.6 Subroutines `rdrstrt.ftn` and `wrrstrt.ftn`

As `c_cplg_step` is not used anymore, it was removed from the restart writing and reading.

[5] Valid points must be equal to 1.0, non valid point to 0.0

4.1.7 Subroutine c_set_coupling.ftn: coupling mask definition and OASIS3-GOSSIP2 initialization

Subroutine c_set_coupling.ftn defines the OASIS coupling mask and the mask sent as output coupling field, and performs OASIS3-GOSSIP2 coupling initialization steps.

Definition of OASIS coupling mask and mask output coupling field

If a coupling mask is defined as a geophysical field by the user (see 4.1.5), this array is used as is to define OASIS coupling mask (in integer array ila_msk) and the mask output coupling field (in real array rga_msk). If not (lg_mcpl is false), this definition is based on the sea-land mask and on MEC coupling role with the following rules[6]:

- If MEC is "atmos", or "ocean" and "land", the whole domain is valid, or
- if MEC is "ocean", grid meshes with at least a fraction of water are valid, or
- if MEC is "land", grid meshes with at least a fraction of land are valid.

OASIS3-GOSSIP2 initialization

If OASIS3-GOSSIP2 is used to couple MEC to other models (namelist key C_coupling⊥=.true.), some coupling initialization is then performed by calling c_oasis_init.ftn90 (see section 3.3.1) and c_init_oasisfld.ftn.

Routine c_oasis_init.ftn90 calls the standard PSMILe routines to perform the general coupling initialization (prism_init_comp_proto) and grid declaration (prism_start_grids_writing, prism_write_grid, prism_write_mask and prism_terminate_grids_writing).

Routine c_init_oasisfld.ftn90 performs the partition declaration (by calling c_oasis_partition.ftn90 which calls prism_def_partition), the declaration of all input and ouput coupling fields (by calling prism_def_var_proto for each field) and the end of the declaration phase (by calling prism_enddef_proto.

For more details about the PSMILe interface, please refer to (5). The initialization and declaration phase is then complete and the model can start it timestep loop.

4.1.8 Subroutines c_fillbus.ftn and c_getbus.ftn: interaction with the physics

Within a timestep, routine c_fillbus.ftn is called by p_physlb.ftn to transfer to the physics the input coupling fields received at the previous timestep (see 4.1.9). The call is done only for and after the third coupling timestep if the run is not a restart, or at each timestep if the run is a restart (i.e. if lg_get=.true.) (see also 5.1 and 4.1.9). In this routine, the input coupling fields are, for each process, in the local array rga_cpl2phy. If a field is a coupling field for the run (i.e. index iga_n_xxx(1) .gt.0), the field is injected in the corresponding dynamic, volatile or permanent bus (F_busdyn, F_busvol or F_busper). Note that c_fillbus.ftn and c_getbus.ftn are called, as the physics, slice per slice for the whole process domain ("slab" part of the code).

Extra space was created in the dynamic bus F_busdyn to include the input coupling fields containing the mask, the surface sensible heat flux, and the surface latent heat flux over the four types of surface (land: MCL, FCL, FVL; glaciers: MCG, FCG, FVG; water: MCW, FCW, FVW; and ice MCI, FCI, FVI) (see also 4.2.4). These fields affect the physics in routines glaciers1.ftn, isba2.ftn, seaice1.ftn, and water.ftn (see 4.2.5). In the dynamics, the index of these fields in F_busdyn (for the first type

[6]Unfortunately, OASIS has a mask convention opposite to the usual one: valid points are identified by an integer value equal to 0, non valid point by an integer value equal to 1.

of surface) are given by integers `mccpl`, `fccpl`, and `fvcpl` retrieved by routine `inikey.ftn` (see 4.1.4).

Within a timestep, the physics is then performed and routine `c_getbus.ftn` is called to extract from the physics the surface fields, present in the dynamic, volatile or permanent bus (`F_busdyn`, `F_busvol` or `F_busper`), that will be sent as output coupling fields or transfered to other modules . These fields fill the process local array `rga_cpl2phy` and will be later collected in `c_cplg_surf.ftn` (see 4.1.9).

4.1.9 Subroutine `c_cplg_surf.ftn`

The coupling to other models via OASIS3-GOSSIP2, or to other modules directly included, is done in `c_cplg_surf.ftn`.

The coupling fields extracted from the physics are collected on the master process into array `rga_fldout`. After this step, any module could be called in `c_cplg_surf.ftn` by the master process with `rga_fldout` as argument. Note that the index of the first element of an output or input coupling field XXX is given by index `iga_xxx(2)` or `iga_xxx(1)` in `rga_fldout` or `rga_fldin` respectively.

The mask output coupling field is then defined based on array `rga_msk` (see 4.1.5 and 4.1.7).

Finally the master process sends the output coupling fields by calling `c_oasis_put.ftn90` which calls PSMILe routine `prism_put_proto`, and receives the input coupling fields by calling `c_oasis_get.ftn90` which calls `prism_get_proto`. Note that those coupling routines are called at each timestep, but that the output or input actions are effectively automatically performed below the `prism_put` and `prism_get` only at the coupling frequency defined by the user in the OASIS configuation file *namcouple*. When a coupling field is effectively received, it is then distributed to the other processes in local array `rga_cpl2phy`.

In this routine:

- integer `ig_ncpl` defines the number of times coupling fields were received;
- logical `lg_getfirst` determines if valid coupling fields were received for the first time in the current timestep (see also 5.1); this will trigger the transfer of `iga_cplfld` to the physics, see 4.2.3;
- logical `lg_get` determines if coupling fields were received in the current timestep and should therefore be sent to the physics with `c_fillbus.ftn`.

4.1.10 Subroutine `c_stop_coupling.ftn`: coupling termination

After MEC has done all its timesteps, it calls routine `c_stop_coupling.ftn` which calls PSMILe routine `prism_terminate_proto` to terminate the coupling.

4.1.11 Subroutines `c_cplg_iml.ftn` and `c_init.ftn`

Routines `c_cplg_iml.ftn` and `c_init.ftn` were removed.

4.2 Modifications in the physics

The figure 4.4 lists the routines modified in the physics part of the code. The modifications brought to those routines are described below in the same order.

4.2.1 Subroutine `phyopt.ftn`

This routine initializes logical `COUPLING` (from `options.cdk`) to `.false.` and receives its value for the run transfered from the dynamics by routine `p_config.ftn` (see 4.1.3). This logical identi-

```
phyopt.ftn
options.cdk

phydebu4.ftn

phycom.ftn

iniptsurf.ftn
phy_ini.ftn
sfcbus.cdk
phybus.cdk

glaciers1.ftn
isba2.ftn
seaice1.ftn
water.ftn
```

Figure 4.4: Modified routines and comdecks in the physics of MEC code

fies whether or not MEC is coupled to other models and therefore whether or not the fluxes should be overwritten by the flux coupling fields (see 4.2.5).

4.2.2 Subroutine `phydebu4.ftn`

This routine checks that the coupling option (`COUPLING`) and the implicit flux boundary condition (`IMPFLX`) are not both true at the same time. This is not allowed as the overwriting of the fluxes by the flux coupling fields (see 4.2.5) would then not be possible.

4.2.3 Subroutine `phycom.ftn`

Routine `phycom.ftn` is called once by `p_phystep.ftn` after valid coupling fields were received for the first time (`lg_getfirst=.true.`, see 4.1.9 and 5.1) to transfer `iga_cplfld`, the array of integer identifying which field is a coupling field in current simulation (see 4.1.2). This ensures that in the physics the fluxes are overwritten by the flux coupling fields only after valid coupling fields are received.

4.2.4 Routines `phy_ini.ftn` and `iniptsurf.ftn`

Those routines were modified to add extra space to include the input coupling fields containing the mask (MCL, MCG, MCW, MCI), the surface sensible heat flux (FCL, FCG, FCW, FCI), and the surface latent heat flux (FVL, FVG, FVW, FVI) over the four types of surfaces in the dynamic bus `F_busdyn` at the addresses `mccpl`, `fccpl`, and `fvcpl` respectively.

4.2.5 Routines `glaciers1.ftn`, `isba2.ftn`, `seaice1.ftn`, and `water.ftn`

Those 4 routines were modified in the same way. If the coupling mode is on (`COUPLING=.true.`), and if the sensible heat flux and the latent heat flux are valid coupling fields, the corresponding fluxes are overwritten with the input coupling fluxes for the non-masked grid points given by the input coupling mask.

4.3 Adaptation of MEC namelist

New namelist entries were introduced in the `&coupling` part of MEC namelist:

- `C_coupling_L`: if `.true.`, means that coupling information is extracted from the physics or fed back to the physics. This key must of course be true to couple MEC with other models via the OASIS3-GOSSIP2 coupler but also to include other modules in the current MEC code (see also 4.1.9).
- `C_cploasis_L`: if `.true.`, means that MEC is coupled with other models via the OASIS3-GOSSIP2 coupler.
- `C_cpld2ocean_L`: if `.true.`, means that MEC is coupled to an ocean model.
- `C_cpld2land_L`: if `.true.`, means that MEC is coupled to a land model.
- `C_cpld2atmos_L`: if `.true.`, means that MEC is coupled to an atmos model.
- `lg_atmos`: if `.true.`, forces MEC to play an "atmos" coupling role.
- `lg_ocean`: if `.true.`, forces MEC to play an "ocean" coupling role.
- `lg_land`: if `.true.`, forces MEC to play an "land" coupling role.
- `from_atmos_S`: redefines "from_atmos" set of coupling fields if the user does not want to use the default definition (see 4.1.2); use `'NIL'` to explicitely define an empty set.
- `from_land_S`: redefines "from_ocean" set of coupling fields if the user does not want to use the default definition (see 4.1.2); use `'NIL'` to explicitely define an empty set.
- `from_ocean_S`: redefines "from_land" this set of coupling fields if the user does not want to use the default definition (see 4.1.2); use `'NIL'` to explicitely define an empty set.

The keys `C_cpld2ocean_L`, `C_cpld2land_L`, and `C_cpld2atmos_L` define to which type of other model MEC is coupled and therefore determine the sets of input coupling fields (see also 4.1.2).

The keys `lg_atmos`, `lg_ocean` and `lg_land` overwrite default MEC coupling role and therefore define the sets of output coupling fields (see also 4.1.2).

4.4 Testing the modifications with MECatm-MECsurf coupling

The modifications implemented in MEC were tested coupling two MECs together. To have a light setting, both MECs are run in the offline mode (i.e. `schm_offline_L=.true.` in gem_setting.nml). However one MEC, which we call here MECatm, is forced to play an "atmos" coupling role. Its `&coupling` part in gem_setting.nml is as follows:

```
&coupling
 C_coupling_L=.true.,
 C_cploasis_L=.true.,
 C_cpld2ocean_L=.true.,
 C_cpld2land_L=.true.,
 C_cpld2atmos_L=.false.,
 lg_atmos = .true.,
 lg_ocean = .false.,
 lg_land  = .false.,
 from_atmos_S = 'TJA', 'UDA', 'VDA', 'TDA', 'FBA', 'FIA', 'RTA',
 from_land_S =  'MCL', 'FCL', 'FVL',
 from_ocean_S =  'MCW', 'FCW', 'FVW', 'GLW', 'I8W', 'I7W', 'TMW',
```

The other MEC, which we call here MECsurf, was naturally playing the "ocean" and "land" roles. Therefore, its `&coupling` part in gem_setting.nml is as follows:

```
&coupling
```

```
C_coupling_L=.true.,
C_cploasis_L=.true.,
C_cpld2ocean_L=.false.,
C_cpld2land_L=.false.,
C_cpld2atmos_L=.true.,
from_atmos_S = 'TJA', 'UDA', 'VDA', 'TDA', 'FBA', 'FIA', 'RTA',
from_land_S =  'MCL', 'FCL', 'FVL',
from_ocean_S = 'MCW', 'FCW', 'FVW', 'GLW', 'I8W', 'I7W', 'TMW',
```

With this setting, MECatm receives the sets of coupling fields "from_land" and "from_ocean" and sends "from_atmos", and vice-versa for MECsurf. As those two models have the same grid, no interpolation is required and the `NOINTERP` transformation is chosen in OASIS configuring file `namcouple` file; this file is given in Appendix A.

Those two MECs were coupled for 6 timesteps on the Linux station `armc28`, which validated the technical exchange of the coupling fields. This first step must now be followed by more extensive validation tests, in particular to ensure the validity of the output coupling fields extracted from the physics and to verify that the feedback to the physics is working properly (see also 5.3).

Chapter 5

Coupling MEC with GOM ocean model

5.1 Coupling algorithm

The coupling algorithm between MEC atmosphere model and GOM ocean model is illustrated in figure 5.1.

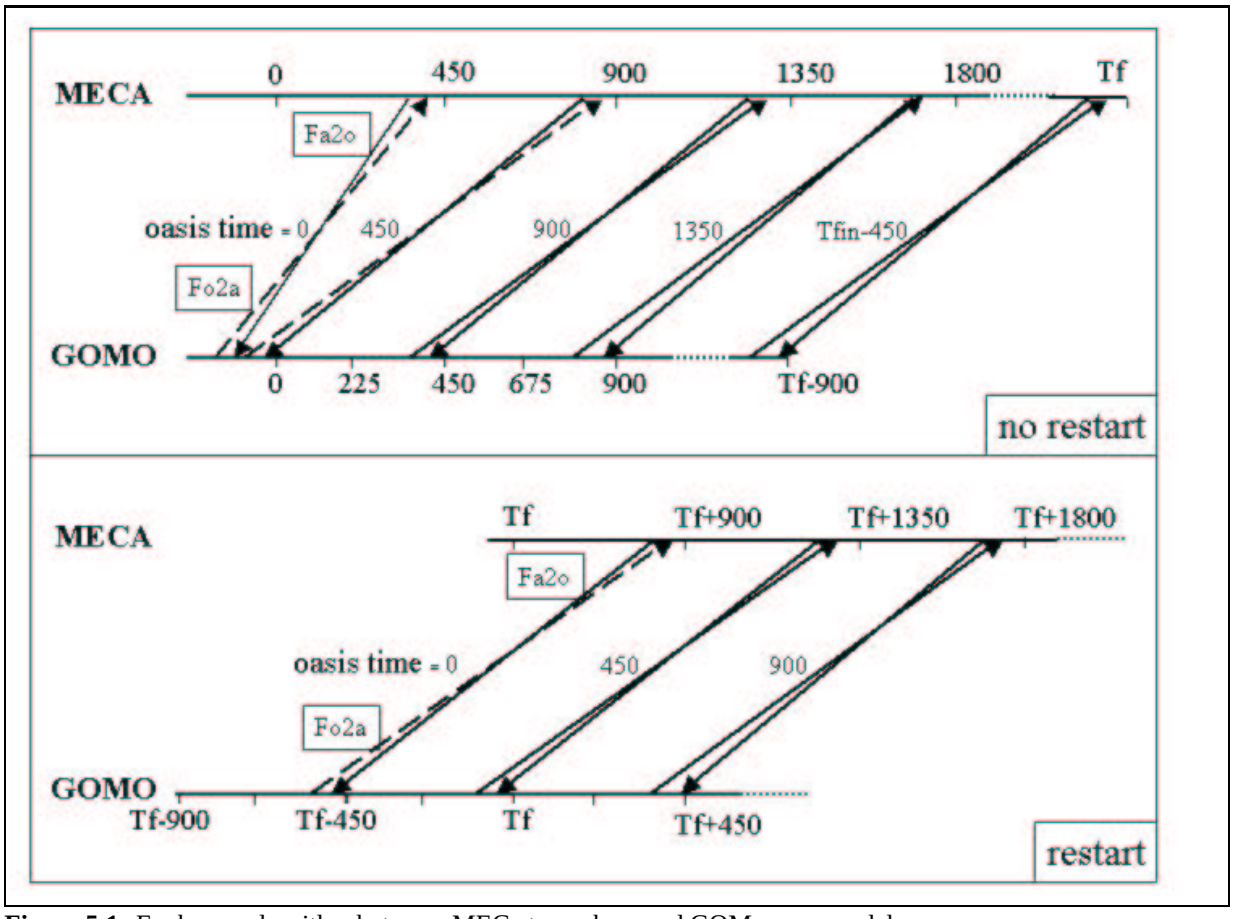

Figure 5.1: Exchange algorithm between MEC atmosphere and GOM ocean models

The situation differs whether the run is a restart or not.

When the run is not a restart run (top pannel in figure 5.1), the ocean model GOM performs two coupling exchanges before entering its timestep loop. Then, as the coupling period is 450 seconds and its timestep only 225 seconds, it performs a coupling exchange at the end of every other timestep[1]. The atmosphere

[1] In fact, the `prism_put_proto` and `prism_get_proto` routines are called every timestep in the code, but the exchange

model MEC simply performs a coupling exchange at the end of every timestep as its timestep is equal to the coupling period, i.e. 450 seconds; but MEC has to ignore the coupling data received by GOM at times 450 and 900 as those data provided by GOM before its timestepping loop are not valid coupling fields; this is represented by dashed arrays on figure 5.1 (see also 4.1.9).

If the run is a restart run (bottom pannel in figure 5.1), GOM simply performs a coupling exchange at the end of every other timestep and MEC at the end of every timestep.

In both cases (restart or not), this coupling algorithm implies that GOM uses for a period extending from T to $T + 450$ the information coming from the interpolation of coupling fields provided by MEC at $T + 450$ and $T + 900$; there is therefore a shift of one coupling period. On the other hand, MEC uses, for the period extending from T to $T + 450$ coupling fields provided by GOM at $T - 900$; there is therefore a shift of two coupling periods. Furthermode, the runs finish at a certain time T_f for MEC and $T_f - 900$ for GOM.

5.2 Coupling fields

The coupling fields send from GOM to MEC, F_{o2a} on figure 5.1, and the 3-letter acronyms used in the code, are the following:

- sea ice fraction (GLW)
- sea ice thickness (I8W)
- sea ice temperature (I7W)
- sea surface temperature (TMW)
- surface sensible heat flux over water (FCW)
- latent heat flux over water (FVW)
- mask over water (MCW)
- surface sensible heat flux over ice (FCI)
- latent heat flux over ice (FVI)
- mask over ice (MCI)

The first 4 fields (GLW, I8W, I7W, and TMW) are passed to the atmosphere as diagnostics and are not really used in MEC. As GOM includes also a sea ice model, MEC expects to receive from it the surface sensible heat flux, the latent heat flux and the mask over water and ice. In practice, GOM passes the same surface sensible heat flux (surface averaged over ice and water) for FCW and FCI and the same latent heat flux (surface average over ice and water) for FVW and FVI, and the same sea-land mask for MCW and MCI. Flux conservation is ensured as those fluxes are used, in MEC, to overwrite the water fluxes (in `water.ftn`) and the sea ice fluxes (in `seaice1.ftn`) and are later aggregated (in `agrege.ftn`). Note that with this configuration, the sea and water fractions used to do the aggregation are not important as GOM passes twice the same fluxes; if the FCW/FVW and FCI/FVI were really the surface sensible heat flux / latent heat flux over water and ice, it would then be essential, to ensure conservation in the aggregation process, to use the water fraction given by the coupling field GLW, and the ice fraction given by (1-GLW)..

The coupling fields send from MEC to GOM, F_{a2o} on figure 5.1, and the 3-letter acronym used in the code, are expected to be:

- screen level temperature (TJA)
- screen level zonal component of wind (UDA) (see also 5.3)
- screen level meridional component of wind (VDA) (see also 5.3)
- dew point temperature (TDA)
- visible flux towards the ground (solar, shortwave) (FBA)

is automatically activated at the coupling frequency equal to 450 seconds defined in the configuration file namcouple.

- infrared energy flux towards the ground (longwave) (FIA)
- total precipitation rate (RTA)

5.3 Remaining issues in MEC-GOM coupling

During the visit, MEC and GOM component models were technically coupled using the new OASIS3-GOSSIP2 coupler. The work to interface GOM with the OASIS PSMILe library and the final assembling were mainly realized by Manon Faucher, based on the work described in this report. The full validation of the set-up and the refinement of the coupling options were still needed. The following issues were identified as deserving special attention in the on-going developments.

1. OASIS interpolations

   The MEC-GOM coupled model currently uses only the nearest-neighbour OASIS interpolation to transform the coupling fields from the MEC grid to the GOM grid and vice-versa. Conservative interpolation (SCRIP/CONSERV in OASIS) has to be used for the flux transformations in order to ensure conservation. The SCRIP/CONSERV interpolation between the MEC and GOM grid was tested and was found to converge only if the SCRIP library was compiled with 8-byte real precision. The problem arises from the fact that OASIS has to be compiled with 4-byte real precision in order to exchange 4-byte coupling fields. There are two possible ways to solve this problem:

   (a) Modify the SCRIP library to ensure that it runs with 8-byte real variables even if OASIS general precision is 4 bytes for real variables. This would be the cleanest way but is certainly not straightforward as many real variables are passed from OASIS to the SCRIP library.

   (b) Use OASIS compiled with 8-byte real precision in the interpolator only mode to calculate offline the weights and addresses of the SCRIP/CONSERV transformation from the MEC to GOM grid and vice-versa; those weights and addresses are automatically stored in a file. Then write a program that reads the 8-byte real weights, transform them into 4-byte reals and write them back into a new file. That new file can then be used by the SCRIP library in OASIS, all compiled in 4-byte real precision. This is probably the easiest way to solve the problem but involves an offline calculation first.

2. Full validation of MEC interfacing modifications

   - The MECatm-MECsurf coupled model was run only for a few timestep to technically validate the coupling modifications. It should however be run for a longer period to fully validate the modifications, in particular the feedback of the coupling fields in the physics.
   - It should also be noted that even if all the modifications were done taking into account the parallelism in MEC, the coupling of a parallel MEC was never done and therefore still has to be validated.
   - The coupling fields 'TJA', 'TDA', 'FBA', 'FIA', 'RTA' extracted in `c_getbus.ftn` from the physics for the MEC-GOM coupling were visually inspected and validated. All the other coupling fields still have to be verified.
   - The coupling fields 'UDA' and 'VDA' should be the screen level zonal and meridional components of the wind. It still has to be checked whether or not the coupling fields extracted from the physics are effectively the zonal and meridional components or if they are the components along the local X and Y directions of the grid. In the latter case, a rotation would have to be done before sending out those fields in `c_cplg_surf.ftn`.

3. Mask issue

   Currently, the mask used to identified the region that must be overwritten by the flux coupling fields in the physics comes from the mask input coupling fields "MCW" and "MCI". For coherency, this should probably be changed and the user-defined coupling mask `MCPL` contained in array `rga_msk` (after reading by subroutine `readgeo.ftn`, see 4.1.5) should probably be globally distributed in

`c_cplg_surf.ftn` (see 4.1.9) and put in the dynamic bus `F_busdyn` at the index `mccpl` in routine `c_fillbus.ftn` (see 4.1.8).

4. Restart mechanism

   The restart mechanism was not implemented. At the end of a run the coupling fields just received should be stored. At the beginning of a restart run, they should be available to run the first coupling period at the end of which coupling fields are then received and updated (see figure 5.1).

5. In the OASIS configuration file `namcouple`, the total time `$RUNTIME` has to be one coupling period greater than the actual run time, so that all coupling exchanges are effectively performed. The reason for this still has to be investigated.

6. Porting of the coupled model on SGI and IBM platforms at RPN still has to be done. Note that the OASIS3-GOSSIP2 coupler was already run on the SGI `pollux` platform but not on the IBM (see also 3.2).

7. The phasing between the final version of the modified MEC routines and GEM 3.2.1 still needs to be done (see chapter 4).

8. For particular issues about the OASIS3-GOSSIP2 coupler, please see sections 3.1.2, 3.2.2, and 3.3.2.

Chapter 6

Conclusions

At the end of the visit, a coupled model based on GEM and IML ocean model was technically assembled using the new OASIS3-GOSSIP2 coupler. The full validation of the set-up, the refinement of the coupling options, and the porting on the IBM Power4 platform were still needed and are currently underwork at RPN.

Regarding the joined coupler developments, the RPN GOSSIP2 communication layer is fully validated into OASIS3 and will be available to the OASIS user community in the next official version that should be released beginning of 2006. The success of this first year of interaction prefigures the establishment of an active and stable collaboration between RPN and CERFACS.

Appendix A

The MECatm-MECsurf namcouple file

The OASIS3-GOSSIP2 configuration file *namcouple* used for the MECatm-MECsurf coupling is as follows:

```
# Any line beginning with # is ignored. Blank lines are not allowed.
#
 $SEQMODE
# This keyword concerns the coupling algorithm. Put here the maximum number
# of fields that have to be, at one particular coupling timestep,
# exchanged sequentially in a given order.
   1
 $END
#############################################################################
 $CHANNEL
# The communication technique you want to use.
# Choices are MPI1 or MPI2, NONE, GSIP
# - if you want to use MPI1 or MPI2 message passing, you must write
# MPI1 or MPI2 (+ NOBSEND if you do not want to use the default MPI_BSend)
# on one line
# + one line per model giving for the model the total number of procs,
# the number of procs implied in the coupling and, for MPI2 only, an
# optional launching argument
#
  GSIP
   1 1
   1 1
 $END
#############################################################################
 $NFIELDS
# This is the total number of fields being exchanged.
# For the definition of the fields, see under $STRINGS keyword
#
          17
 $END
#############################################################################
 $JOBNAME
# This is an acronym for this run (3 characters).
  MEC
 $END
#############################################################################
 $NBMODEL
# This gives you the number of models running in this experiment +
# their names (6 characters) + , in option, the maximum Fortran unit
# number used by each model; 1024 will be used if none are given.
#
  2  MECatm MECloc
 $END
#############################################################################
 $RUNTIME
# This gives you the total simulated time for this run in seconds
#
  2700
 $END
#############################################################################
 $INIDATE
# This is the initial date of the run. This is important only if
# FILLING analysis is used for a coupling field in the run.
```

```
# The format is YYYYMMDD.
  00010101
 $END
#############################################################################
 $MODINFO
# Indicates if a header is encapsulated within the field brick
# in binary restart files for all communication techniques,
# (and for coupling field exchanges for PIPE, SIPC and GMEM.
# (YES or NOT)
  NOT
 $END
#############################################################################
 $NLOGPRT
# Index of printing level in output file cplout: 0 = no printing
#  1 = main routines and field names when treated, 2 = complete output
  2
 $END
#############################################################################
 $CALTYPE
# Calendar type :  0      = 365 day calendar (no leap years)
#                  1      = 365 day, or 366 days for leap years, calendar
#                  n (>1) = n day month calendar
# This is important only if FILLING analysis is used for a coupling
# field in the run.
#
  30
 $END
#############################################################################
 $STRINGS
#
# The above variables are the general parameters for the experiment.
# Everything below has to do with the fields being exchanged.
#
#############################################################################
#                      MECatm  --->>>  MECloc
#                      ----------------------
# Field 1
#   First line:
# 1) and 2) Symbolic names for the field in the source model and target model
# 3) Index of field in cf_name_table.txt
# 4) Exchange frequency for the field in seconds
# 5) Number of analysis to be performed
# 6) Restart input NetCDF file names
# 7) Field status
#
MECATJA MECLTJA 206 900  1  dummy.nc    EXPORTED
#
#   Second line:
# 1) and 2) grid locator prefix
# + possibly LAG, SEQ, DEL, XTS additional indices if needed
#
MECa MECl
#
#   Third line:
# Source grid characteristic (P or R) and number of overlapping grid points
# + Target grid characteristic (P or R) and number of overlapping grid points
R  0  R 0
#
# List of analyses
#
NOINTERP
#
# Field 2
#
MECAUDA MECLUDA 206 900  1  dummy.nc    EXPORTED
MECa MECl
R  0  R 0
NOINTERP
#
# Field 3
#
MECAVDA MECLVDA 206 900  1  dummy.nc    EXPORTED
MECa MECl
R  0  R 0
NOINTERP
#
# Field 4
```

```
#
MECATDA MECLTDA 206 900  1  dummy.nc    EXPORTED
MECa MECl
R  0  R 0
NOINTERP
#
# Field 5
#
MECAFBA MECLFBA 206 900  1  dummy.nc    EXPORTED
MECa MECl
R  0  R 0
NOINTERP
#
# Field 6
#
MECAFIA MECLFIA 206 900  1  dummy.nc    EXPORTED
MECa MECl
R  0  R 0
NOINTERP
#
# Field 7
#
MECARTA MECLRTA 206 900  1  dummy.nc    EXPORTED
MECa MECl
R  0  R 0
NOINTERP
#
#########################################################################
#                       MECloc --->>>  MECatm
#                       ---------------------
#
# Field 8
#
MECLMCL MECAMCL 383 900  1   dummy.nc   EXPORTED
MECl MECa
R  0  R 0
NOINTERP
#
# Field 9
#
MECLFCL MECAFCL 383 900  1   dummy.nc   EXPORTED
MECl MECa
R  0  R 0
NOINTERP
#
# Field 10
#
MECLFVL MECAFVL 383 900  1   dummy.nc   EXPORTED
MECl MECa
R  0  R 0
NOINTERP
#
# Field 11 : masque
#
MECLMCW MECAMCW 206  900  1   dummy.nc   EXPORTED
MECl MECa
R  0  R 0
NOINTERP
#
# Field 12 : Sensible heat flux
#
MECLFCW MECAFCW 384  900  1   dummy.nc   EXPORTED
MECl MECa
R  0  R 0
NOINTERP
#
# Field 13 : Latent heat flux
#
MECLFVW MECAFVW 383 900  1   dummy.nc   EXPORTED
MECl MECa
R  0  R 0
NOINTERP
#
# Field 14
#
MECLGLW MECAGLW 383 900  1   dummy.nc   EXPORTED
MECl MECa
```

```
R  0  R 0
NOINTERP
#
# Field 15
#
MECLI8W MECAI8W 383 900  1   dummy.nc   EXPORTED
MECl MECa
R  0  R 0
NOINTERP
#
# Field 16
#
MECLI7W MECAI7W 383 900  1   dummy.nc   EXPORTED
MECl MECa
R  0  R 0
NOINTERP
#
# Field 17
#
MECLTMW MECATMW 383 900  1   dummy.nc   EXPORTED
MECl MECa
R  0  R 0
NOINTERP

#############################################################################
#
 $END
```

Bibliography

[1] http://prism.enes.org

[2] Desjardins, S. 2003: OASIS vs RPN-Coupler. MSC internal report.

[3] Legutke, S. and V.Gayler, 2004: The PRISM Standard Compilation Environment. PRISM Report Series No 4.

[4] Gayler, V. and S. Legutke, 2004: The PRISM Standard Running Environment. PRISM Report Series, No 5.

[5] Valcke, S., A. Caubel, R. Vogelsang, and D. Declat, 2004: OASIS3 User Guide (oasis3_prism_2-4). PRISM Project Report No 2, 5th Ed., 60 pp.

[6] Valcke, S., 2004: The PRISM TOYCLIM Coupled Model Adaptation Guide. PRISM Report Series, No 7, 70 pp.

[7] Bouhemhem, D., 2004: GOSSIP: Globally Organized System for Simulation Information Passing. MSC internal report.