Coupling BFM with ocean models:

the OGSTM model

P. Lazzari, G. Bolzon

Release 1.0, February 2023
—— BFM Report series N. 4 ——

http://bfm-community.eu
bfm_st@cmcc.it

BFM
BIOGEOCHEMICAL FLUX MODEL

This document should be cited as:
Lazzari, P., Bolzon, G. (2023) Coupling BFM with OGSTM model. BFM Report series N. 4, Release 1.0, February 2023, Trieste, Italy, http://bfm-community.eu

Contents

1 Introduction

1.1 Basic equations of the coupling

The BFM-OGSTM coupled software is essentially a numerical solver for the transport reaction equations where the reaction terms are defined by the BFM framework.

The equations solved are a system of partial differential equations of the form:

$$\frac{\partial C}{\partial t} = -\mathbf{u}\cdot\nabla C + \nabla_H\cdot(A_H\nabla_H C) + \frac{\partial}{\partial z}A_V\frac{\partial C}{\partial z} - w_B\frac{\partial C}{\partial z} + \left.\frac{\partial C}{\partial t}\right|_{bio} \tag{1.1.1}$$

where $\mathbf{u}\equiv(u,v,w)$ is the three-dimensional current velocity and (A_H, A_V) are the horizontal and vertical turbulent diffusivity coefficients. w_b is the sinking velocity of living or dead organic particulate matter. The bio derivatives are the reaction terms that are expressed as a set of ordinary differential equations that express the transformation fluxes of biogeochemical properties in the marine environment. The BFM formualtion and the standalone implementation are described in full details in the BFM manual (Vichi et al., 2023).

In the present version OGSTM uses a Smolarkiewicz scheme to solve advection and a bilaplacian scheme for horizontal diffusion. Advection, horzointal diffusion and biological transformations are integrated in time using an euler forward time step while vertical diffusion is solved by means of an implicit scheme.

In the following the model containing the main program file and responsible for the spatial and temporal intregration is referred as the host model.

The code of the ogstm is organized in a number of subdirectories each containing subroutines and modules related to specific aspects of the code:

```
BC      BIO     DA      General IO      MPI     PHYS    namelists
```

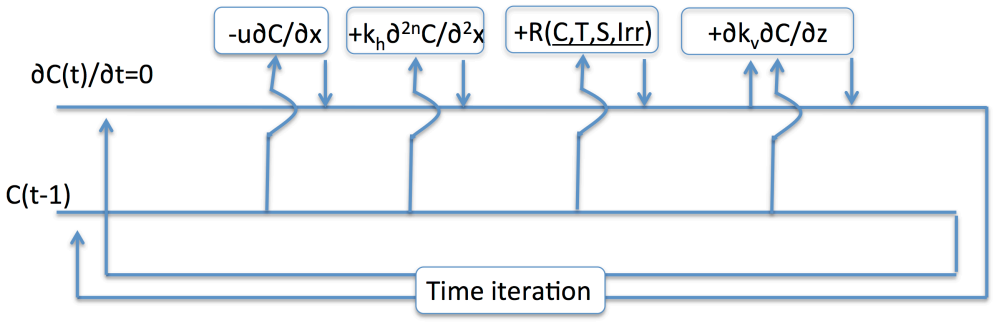

Figure 1.1: Scheme of the numerical integration of the OGSTM model.

the coupling elements are presents in the General/ directory for the initialization and finalization files, and in the BIO directory for the stepping phase.

1.2 Coupling in terms of coding

The approach used to couple the pelagic component of the BFM is a 1D approach, based on sea water columns. In particular the BFM is invoked as an external library totally independent from the host three-dimensional model. The coupling is realized in each of the three principal sections of the code:

1. the initialization routine;
2. the step routine;
3. the finalization routine.

The BFM initialization calls are in the ogstm_initialize subroutine inside General/ogstm.f90:

```
! Initialization of Biogeochemical reactor with 1D approach

      call BFM1D_NO_BOXES(jpk,1,1,jpk,1)
      parallel_rank=myrank
      call Init_bfm()
      call BFM1D_INIT_IO_CHANNELS()
      call Initialize()
```

The BFM1D_NO_BOXES() subroutine essentially tells to the BFM library that it has to allocate a one dimensional spatial box. The BFM1D_INIT_IO_CHANNELS() initialize the IO for the ascii BFM files. The Initialize() subroutine allocates the internal memory of the BFM library. In the top call, the ogstm() subroutine, an *use* statement including all the BFM modules (#include "BFM_module_list.h") is present in order to force the consistency of allocated memory in all the following subroutines called in the algorithm tree.

The main part of the coupling is in the “step” phase, in the src/BIO/trcbio.f90 subroutine, the following pieces of code are inside a loop on ji, jj index (zonal and meridional indexes). As shown in the panel the loop on tracers index, jtr, put all the tracer state vector for a single column of the 3D domain in the two dimensional vector a. Moreover the environmental regulator vector (er) is assigned with the values of the parameters needed by BFM.

```
      DO jtr=1, jtrmax
           a(1:bottom, jtr) = trn(1:bottom,jj,ji,jtr) ! current biogeochemical
               concentrations
      END DO

! Environmental regulating factors (er)
er(1:bottom,1)  = tn (1:bottom,jj,ji)      ! Temperature (Celsius)
er(1:bottom,2)  = sn (1:bottom,jj,ji)      ! Salinity PSU
er(1:bottom,3)  = rho(1:bottom,jj,ji)      ! Density Kg/m3
er(1       ,4)  = ice                      ! from 0 to 1 adimensional
er(1       ,5)  = ogstm_co2(jj,ji)         ! CO2 Mixing Ratios (ppm)
do jk=1, bottom
  er(jk,6) = instant_par(COMMON_DATEstring,xpar(jk,jj,ji))  ! PAR umoles/m2/s |
      Watt to umoles photons W2E=1./0.217                          enddo
er(1       ,7)  = DAY_LENGTH(jj,ji)        ! fotoperiod expressed in hours
er(1:bottom,8)  = e3t(1:bottom,jj,ji)      ! depth in meters of the given cell
er(1       ,9)  = vatm(jj,ji)              ! wind speed (m/s)
```

```
er(1:bottom,10) = ogstm_PH(1:bottom,jj,ji) ! 8.1
do jk=1,bottom
   correct_fact= 1.0D0
   if ( (gdept(jk,jj,ji) .GT. 1000.0D0 ) .AND.  (gdept(jk,jj,ji) .LT. 2000.0D0))
       then
     correct_fact= 0.25D0
   endif
   if (gdept(jk,jj,ji) .GE. 2000.0D0 ) then
     correct_fact= 0.0D0
   endif
   er(jk,11) = correct_fact * ( gdept(jpk,jj,ji)-gdept(jk,jj,ji) ) /gdept(jpk,jj,
       ji)
enddo
```

the tracer state vector *a* and the vector *er* are transferred to the internal memory of the BFM library by means of the function BFM1D_Input_EcologyDynamics(). The BFM1D_reset() subroutine reset all the derivatives in the BFM internal memory.

```
call BFM1D_Input_EcologyDynamics(bottom,a,jtrmax,er)

call BFM1D_reset()

call EcologyDynamics()

call BFM1D_Output_EcologyDynamics_surf(b, c, d ,d2)

DO jtr=1, jtrmax
    tra(1:bottom,jj,ji,jtr) =tra(1:bottom,jj,ji,jtr) +b(jtr,1:bottom) ! trend
END DO
```

Then the EcologyDynamics() updates the BFM SOURCE array and in the end the BFM1D_Output_EcologyDynamics() extracts the derivatives (*b*) and other parameters (like sinking velocity *c*) and diagnostics (*d* and *d2* - for surface cells-) defined in the layout of the BFM. The derivatives contained in the vector *b* for the column are added to the tracer matrix *tra* to compute the cumulative derivative as required in the source splitting approach. We recall that all the coupling instructions are perfomed in a loop over the surface wetpoints.

2 Installation, configuration and compilation

2.1 Installation

Both the OGSTM and the BFM codes are under a GIT versioning systems. The installation phase starts with the downloading of the code. There is a public github repository (https://github.com/inogs/ModelBuild) designed to download the codes and build together. This can be done with the following steps:

```
git clone git@github.com:inogs/ModelBuild.git CODE
cd CODE
git checkout testcase

./downloader_ogstm_bfm.sh # downloads bfm and ogstm
# Configure BFM: edit layout and namelists_bfm (fully detailed in the BFM
    manual)
# Now, edit builder_ogstm_bfm.sh
./builder_ogstm_bfm.sh    # builds bfm library and ogstm executable
```

The builder needs to be edited at its top, about compiler flags and machine dependent settings. Here we present, as default, a module file in ogstm/compilers/machine_modules/ section with intel compiler and consistent netcdf libraries.

```
#          builder_ogstm_bfm.sh
#      Edit sections 1,2,3,4 in order to configure compilation and linking.
#################################################################
#  Section 1. Choose the *.inc file, with the definition of compiler flags, to be
     included in Makefile
# #           This is a machine dependent operation, flags for
#             most popular compilers (gnu, intel, xl) are provided.
#             In the following example user will select the file x86_64.LINUX.
    intel.dbg.inc
#             both in bfm/compilers/ and ogstm/compilers
OGSTM_ARCH=x86_64
OGSTM_OS=LINUX
OGSTM_COMPILER=intel
DEBUG=        # this is the choice for production flags
#DEBUG=.dbg   # this is the one for debug flags
#################################################################
#  Section 2. Use of OpenMP threads, to improve the parallelization of ogstm.
# Just comment one of thes lines:
export OPENMP_FLAG=-fopenmp  # OpenMP activated
export OPENMP_FLAG=          # OpenMP deactivated
#################################################################
# Section 3.  Module loads (and set of environment variables)
# This is a machine dependent operation. Modules are usually used on clusters.
# User can write his module file, in the directory below there are some examples.
# Warning : this choice must be consistent with Section 1.
# Just comment the two following lines you are not using modules.
```

```
export MODULEFILE=$PWD/ogstm/compilers/machine_modules/g100.intel
```

2.2 Configuring BFM with OGSTM

Inside builder_ogstm_bfm.sh there are these lines to generate the code and compile the bfm library. The first one edits the file configuration to set the .inc file consitent with compiler we want to use.

Then, there is the actual automatic configuration of the model by using the script *bfm_configure.sh* (see the BFM manual for more information or simply run `./bfm_configure.sh -h`).

```
% sed -i "s/.*ARCH.*/        ARCH    = '$INC_FILE'  /"  build/configurations/
    OGS_PELAGIC/configuration
% cd $BFMDIR/build
% ./bfm_configure.sh -gcv -o ../lib/libbfm.a -p OGS_PELAGIC
```

Every time the layout is changed this procedure must be repeated in order to create updated libraries and then the ogstm executable must also be recompiled.

Here we display the content of build/configurations/OGS_PELAGIC/configuration

`configuration`: compilation and deployment options

This file uses the F90 namelist format to set values and strings must be surrounded by the ' character:

```
&BFM_conf
        MODE    = 'OGS',
        CPPDEFS = 'BFM_STANDALONE INCLUDE_PELCO2 BFM_OGS',
        ARCH    = 'x86_64.LINUX.intel.inc',
        PROC    = 4,
        EXPFILES = 'namelists.passivetrc'
        EXP     = 'ogs.pelagic',
/
```

3 Running BFM OGSTM testcase

To run a test simulation of the code the first thing is to be sure to have a working python3 installed on the machine with the scipy and netcdf module installed.

Then, go in ogstm/testcase/ directory and copy TEST_LIST.dat_template to TEST_LIST.dat and edit the file to configure the needed tests. The tests are performed on a rectangular box with closed boundary conditions.

3.1 Description

The TEST_LIST.dat file contains some of the features of the model domain we want to create, after the header one can add a different specification line for each test she/he wants to create:

```
Nx Ny Nz nprocx nprocy   lon0    lat0     dx      dy        Start           End
             Directory            Code
10 10 43   1      1      8.78125 43.78125 0.128 0.128 20000101-00:00:00
    20010101-00:00:00 TEST01/wrkdir/MODEL ~/TEST_CASE1
...
```

Each line below the header correspond to a different experiment, the default file (TEST_LIST.dat_template) has one experiment: TEST01. Each line specifies a number of features like the number of discretization cells Nx and Ny in the x (longitude) and y (latitude) directions, the number of vertical levels Nz. nprocx and nprocy defines the number of processors to be assigned in the x and y directions for the MPI domain decomposition. lon0 and lat0 indicate the reference corrdinates of the center of the box. “dx” and “dy” indicate the distance of the grid cell centers cordinates. Start and End parameters are the starting and ending dates of the simulation espressed as yyyymmdd-hh:mm:ss. The Directory string defines where to deploy the running environment and the “Code” string indicates where to catch executables and namelists e.g. the root directory of the executable that corresponds to the $OGSTM_HOME directory of the previous section. Ogstm testcase is composed by a suite of python functions invoked in the Main_create_TEST.py caller that creates all the necessary netcdf files and input data to run a biogeochemical simulation:

```
c_mask.create_meshmask_nc(test)
c_bfmmask.create_bfmmask_nc(test)
c_dom.create_Dom_Dec(test)
c_for.create_forcings_nc(test)
c_ext.create_extinction_nc(test)
c_bc.create_bc_nc(test)
create_fluxes.create_fluxes(test)
c_init.create_init_nc(test)
d_code.deploy_code(test)
c_events.create_events(test)
```

the argumet test is a data structure containing all the informations defined in a single line of the TEST_LIST.dat file. Test is iterated along the lines of the TEST_LIST.dat file.

```
c_dom.create_Dom_Dec(test)
```

Create ascii file for MPI domain decomposition (domdec.txt)

```
c_mask.create_meshmask_nc(test)
```

Create meshmask.nc file with Longitude, Latitude, and georeferencing parameters (file meshmask.nc)

```
c_for.create_forcings_nc(test)
```

Create analytical forcing files (temperature, circulation fields, etc etc). The files produced are separated according to four groups (16 of January files are shown):

- Tyyyy0116-12:00:00.nc contains temperature (3D), salinity (3D), wind speed (2D), dilution/concentration field (2D), sea surface height (2D), short wave radiation (2D);
- Uyyyy0116-12:00:00.nc contains zonal velocity (3D);
- Vyyyy0116-12:00:00.nc contains meridional velocity (3D);
- Wyyyy0116-12:00:00.nc contains vertical velocity (3D), and eddy diffusivity (3D).

"yyyy" means that the files are in perpetual year mode. The files are collocated in the directory FORCINGS.

```
c_ext.create_extinction_nc(test)
```

Light extinction parameter files (e.g. KextF_yyyy0107-00:00:00.nc), contains the extinction coefficient parameter for the attenutaion along the water column of photosynthetic available radiation (PAR). The files are in the FORCINGS directory.

```
c_bc.create_bc_nc(test)
```

Boundary condition files (Atmospherical and terrestrial inputs), 4 different types of files

- ATM_yyyy0630-00:00:00.nc Atmospheric deposition of macro nutrients (phosphates and nitrates);
- CO2_yyyy0630-00:00:00.nc CO2 mixing ratio data;
- GIB_yyyy0215-12:00:00.nc Lateral boundary conditions;
- TIN_yyyy0115-00:00:00.nc Terrsetrial inputs (e.g. rivers).

The files for boundary conditions are in the directory BC.

```
c_init.create_init_nc(test)
```

Initial conditions for the tracers initialization, the files are collocated in the RESTART directory.

```
d_code.deploy_code(test)
```

Create a link of the executable form the directory specified in TEST_LIST.dat and copy all the namelists.

```
c_events.create_events(test)
```

Create the event file e.g. output frequency, restart frequency (files 1.aveTimes, 2.aveTimes, restartTimes), by default such files are created empty.

The following command automatically create all the running directory specified in the TEST_LIST.dat file with all the needed files in place.

```
python Main_create_TEST.py
```

3.2 Serial and parallel simulations

The model is consifgured to work in serial or parallel configuration. The serial configuration operates by the following instruction

```
./ogstm.xx
```

The parallel version requires mpirun command and usully the run is submitted by a job, specifing the number of processors and other important aspects of the simulation, we report an example for the slurm scheduler:

```
#!/bin/bash

#SBATCH --job-name=TEST
#SBATCH -N2
#SBATCH --ntasks-per-node=48
#SBATCH --time=24:00:00
#SBATCH --mem=300gb


cd $SLURM_SUBMIT_DIR
module purge
module load ...
module load ...
module load ...
ulimit -s unlimited

date
export RANKS_PER_NODE=48
mpirun -np 96 ./ogstm.xx
date
```

4 Output and diagnostics

Model output is configurable. Output frequency can be defined in the files 1.aveTimes (high frequency) or 2.avetimes (low frequency). The dumping istant is defined by a date (with format yyyymmdd-hh:mm:ss) in the 1.aveTimes file, the output will be the temporal average between dumping instants.

```
20010101-00:00:00
20010201-00:00:00
20010301-00:00:00
20010401-00:00:00
20010501-00:00:00
20010601-00:00:00
20010701-00:00:00
20010801-00:00:00
20010901-00:00:00
20011001-00:00:00
20011101-00:00:00
20011201-00:00:00
20020101-00:00:00
```

the output is constituted by one file per variable per dumping frequency. The model uses the MPI domain decomposition approach, but the master node recombines the domain so the output is indipendent with respect to domain decomposition.

```
...
ave.20010816-12:00:00.B1c.nc
ave.20010816-12:00:00.B1n.nc
ave.20010816-12:00:00.B1p.nc
ave.20010816-12:00:00.N1p.nc
ave.20010816-12:00:00.N3n.nc
ave.20010816-12:00:00.N4n.nc
ave.20010816-12:00:00.N5s.nc
...
```

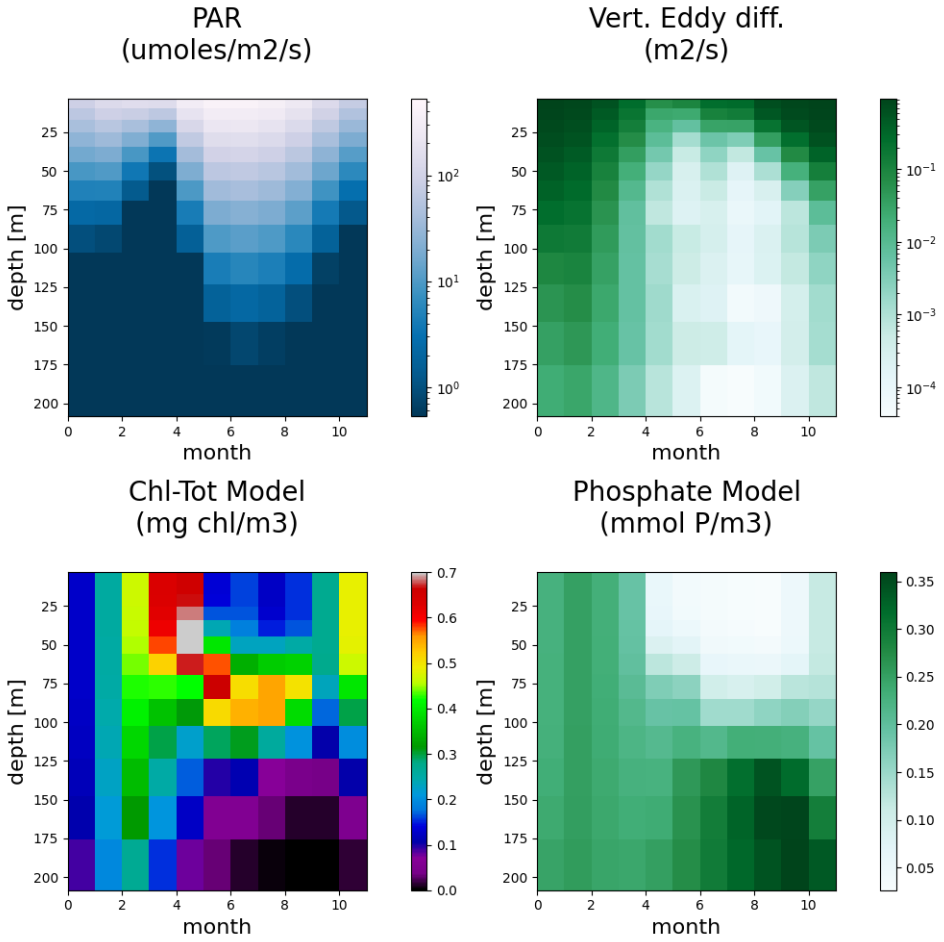

Figure 4.1: Hovmoeller diagrams of model output after one year of simulation. Top left panel shows Photo-synthetical Available Radiation PAR. Top right panel show Vertical eddy diffusivity. Bottom left panel total chlorophyll derived as the sum of the four phytoplankton functional types (TotChl=P1l+P2l+P3l+P4l) used in this BFM configuration. Bottom right panel shows phosphate temporal evolution (N1p).

Bibliography

Vichi, M., Lovato, T., Butenschön, M., Tedesco, L., Lazzari, P., Cossarini, G., Masina, S., Pinardi, N., Solidoro, C., Zavatarelli, M., February 2023. The Biogeochemical Flux Model (BFM): Equation Description and User Manual. BFM version 5.3. BFM Report Series 1, Bologna, Italy.
URL http://bfm-community.eu