The BFM quick-start guide

version 1.4

1. Introduction

The default basic configuration of BFM is the STANDALONE model, which simulates the biogeochemistry of a water volume(s) with given depth(s). The typical example is a micro- or mesocosm batch culture. Both a pelagic and benthic system can be simulated with this configuration. The model is forced with analytical functions of temperature, salinity and light and can be integrated for any desired period with three different numerical schemes. The model output is in NetCDF.

2. Installation

The installation of the model is very simple. Create an installation directory for the BFM and cd into that directory. For example:

```
% mkdir $HOME/BFM
```

```
% cd $HOME/BFM
```

Uncompress and extract the contents of the tarball. The downloaded file will have a different name based on the version. For example:

```
% tar zxvf bfm-<version>.tgz
```

This operation will create and populate the directories `bfm-<version>` containing the source files and bfm-run for running the various test cases.

The full directory tree is given in Appendix A.

3. System Requirements

The currently supported architectures are

- linux
- NEC SX6

Under linux, the model can be currently compiled only with the Intel Fortran compiler (`ifort`). The use of other compilers is underway (*seeking collaborations!*).

A working installation of the NetCDF library (http://www.unidata.ucar.edu/software/netcdf) is mandatory to complete the compilation.

4. Compilation

The user local settings for compilation are provided by means of the following environmental shell variables:

- `FORTRAN_COMPILER`
  The choice is between `IFORT` and `NECSX6`
- `NETCDFINC, NETCDFLIBDIR`
  The paths to the netcdf include and lib directories.
- `BFMDIR`
  The root directory of the BFM model (the path to the installed `bfm-<version>` directory).
- `COMPILATION_MODE`
  Choose between production, debug and profile. Default is production, which optimizes the code execution with the usual optimization flags.

The environmental variables can be set in the user shell configuration file (use the corresponding csh or ba(sh) statements), by editing and executing the provided `bfm_env.(c)sh` script or directly in the file `bfm-<version>/src/Rules.make` which is included in all the `Makefile` (deprecated) .

The compilation is done in the usual way by launching the GNU `make` command in the `src` directory. This will build an executable binary file called `bfm-standalone_prod` in `bin`.

5. Running

The BFM settings can be mostly controlled via namelists. The STANDALONE model behaviour depends on 3 namelist files in the `bfm-run/standalone` directory.

5.1 standalone.nml

This file contains the namelist `standalone_nml` which sets the dimensions of the model, the numerical integration details.

name	*kind*	*description*
`nboxes`	`integer`	Number of water volumes (boxes)
`indepth`	`real`	Depth of each box (m)
`latitude`	`real`	Latitude of each box
`longitude`	`real`	Longitude of each box
`maxdelt`	`real`	Maximum timestep duration (s)
`mindelt`	`real`	Minimum timestep duration (s)
`method`	`integer`	Integration method 1. Euler forward 2. Runge-Kutta 2nd order 3. Leap-frog

The namelist `time_nml` controls the time manager (an extension of the one developed by the GOTM team under the GPL license, http://www.gotm.net).

name	*kind*	*description*
`timefmt`	`integer`	Format of the time limits and loop: 1. Gregorian calendar, MaxN only – give number of iterations, fake start time is used. 2. Gregorian calendar, start and stop dates- MaxN calculated. 3. Gregorian calendar, start and MaxN - stop time calculated. 4. Generic calendar, simdays only– give number of simulation days, fake start time used and MaxN calculated.
`MaxN`	`integer`	Number of iterations in time-loop
`simdays`	`integer`	Number of simulation days
`start`	`string` `"yyyy-mm-dd hh:mm:ss"`	Start date
`stop`	`string` `"yyyy-mm-dd hh:mm:ss"`	Stop date

The namelist `anforcings_nml` controls the variability of the analytical forcings described in Sec. 7.

name	*kind*	*description*
`lw`	`real`	Sinusoidal light intensity (winter) W m^-2
`ls`	`real`	Sinusoidal light intensity (summer) W m^-2
`sw`	`real`	Sinusoidal salinity (winter)
`ss`	`real`	Sinusoidal salinity (summer)
`tw`	`real`	Sinusoidal temperature (winter) degC
`ts`	`real`	Sinusoidal temperature (summer) degC

name	*kind*	*description*
tde	real	Sinusoidal temperature daily excursion degC
botdep_c	real	Organic carbon deposition rate (mg C m^-2 d^-1)
botdep_n	real	Organic nitrogen deposition rate (mmol m^-2 d^-1)
botdep_p	real	Organic phosphorus deposition rate (mmol m^-2 d^-1)
botdep_si	real	Organic silicate deposition rate (mmol m^-2 d^-1)
botox_o	real	Pelagic oxygen ventilation rate (mmol m^-2 d^-1)

5.2 bfm.nml

The namelists contained in this file control the run-time settings such as the configuration (pelagic, benthic), the initial values and the frequency of output and the variables which are stored in the output files.

The major parameters in `bfm_nml` are:

name	*kind*	*description*
bio_setup	integer	BFM configuration: 1. pelagic 2. benthic 3. pelagic and benthic
out_fname	string	Name of NetCDF output file
out_dir	string	Path to the output file
out_title	string	Name of the experiment in NetCDF file
out_delta	integer	Output is saved every `out_delta` timesteps

The namelists `bfm_init_nml, bfm_ben_init_nml` give homogeneous initial values to the state variables (identified with a trailing 0 after the name of the state variable as given in the BFM Description and User Manual).

For example, in the pelagic:

```
n1p0 = 0.75,
n3n0 = 5.0,
```

or in the benthic model:

```
y1c0  = 1600.0,
y2c0  = 470.0,
```

5.3 Param.nml

This is the core of the BFM, where most of the available options can be set. A complete description of the available parameters is given in the User Manual. We report here only the major parameters of interest for the default test cases (see also Sec. 8).

name	*kind*	*description*
`CalcBenthicFlag`	`integer`	Switch for the benthic model 0. no benthic model; 1. simple benthic return; 2. benthic organisms and intermediate complexity nutrient regeneration; 3. benthic organisms and full nutrient regeneration (early diagenesis)
`CalcPhytoplankton(P)`	`logical`	Switch for phytoplankton P=1,2,3,4
`CalcBacteria`	`logical`	Switch for bacteria
`CalcMesoZooplankton(Z)`	`logical`	Switch for mesozooplankton Z=1,2 (Z3,Z4)
`CalcMicroZooplankton(Z)`	`logical`	Switch for microzooplankton Z=1,2 (Z5,Z6)
`CalcPelChemistry`	`logical`	Switch for pelagic chemistry
`CalcBenOrganisms(Y)`	`logical`	Switch for benthic organisms Y=1,2,3,4,5
`CalcBenBacteria(H)`	`logical`	Switch for benthic bacteria H=1,2
`LightForcingFlag`	`integer`	Light forcing options: 1. instantaneous light description 2. constant light over 24h 3. rectangular light distribution (on/off)
`ChlLightFlag`	`integer`	Switch for chlorophyll parameterization in phytoplankton 1. constant chl:C ratio and acclimation through optimal light (ERSEM II) (note: requires the initialization of I_{opt} in phytoplankton, e.g. `P1l` is in W m^{-2}) 2. variable chl:C ratio (note: `P1l` is in mg chl

name	*kind*	*description*
		m^{-3})

6. Numerical integration

Three integration schemes are implemented in the STANDALONE model and can be chosen by the parameter method in the namelist file `standalone.nml`. They are all explicit and use a time step cutting approach to avoid negative concentrations. In detail, they are:

- the Euler scheme(`method=1`):

  $c_{n+1} = c_n + S(c_n)\,\Delta t$

- the Runge-Kutta schem of 2nd order (`method=2`):

  $c_{n+1} = c_n + \frac{1}{2}\left[S(c_n) + S(c_n + S(c_n)\,\Delta t)\right]\Delta t$

- the leap-frog scheme (`method=3`):

  $c_{n+1} = c_{n-1} + S(c_n)\,2\,\Delta t$ with the Asselin-filter: $c_n = c_n + \gamma(c_{n-1} - c_n + c_{n+1})$

7. Analytical forcing functions

For the STANDALONE model a set of simple forcing functions is provided. They are thought of as a first approach to realistic forcing functions to give the user a easily accessible and quickly usable basic set-up of the physical forcing fields.

The forcing can be controlled by the user in adapting the below mentioned parametes which can be changed in the namelist-file `standalone.nml`. The basic concept behind all forcing funcion is the interpolation of some summer and winter value - as the two extreme values – by a sine wave.

7.1 Light forcing

Light forcing is interpolated by a sine wave from a winter extreme value and a summer extreme value (here with their default values in W m^{-2}) to obtain the actual value of average daylight for each day of the year:

```
lw          = 250.0,
ls          = 250.0,
```

The above values are as well intended as the average daylight, i. e. the **mean value of light energy** over the **light period**: $\frac{1}{daylength}\int_{\tau_{dawn}}^{\tau_{dusk}} light(t)\,dt$

In case of the instantaneous light option (`LightForcingFlag=1`) the average daylight is

distributed in a sine wave over the daylight period reaching thus a maximum of twice the average daylight while being zero at the beginning and in the end of the light period.

In the case of constant light (`LightForcingFlag=2`) the mean daylight is smoothed out over the whole day by the factor daylength/24h.

In the case of rectangular light distribution (`LightForcingFlag=3`) the light equals the average daylight value over the whole daylight period and is zero at night time.

Note that for all options of the parameter `LightForcingFlag` in `Param.nml` the amount of light energy in a day's period that is available to the water box is the same.

7.2 Salinity forcing

The salinity forcing is simply given by an interpolation of the extreme values of summer and winter salinity interpolated by a sine wave. The two extreme values and their default values are:

```
sw          = 33.0,
ss          = 37.0,
```

7.3 Temperature forcing

Similar to the salinity forcing the temperature forcing is constructed from a sine wave with a daily mean winter and summer temperature as extreme values. In addition, a daily excursion can be superimposed in sine wave form by imposing its value (constant all over the year). The extreme values and the daily excursion are (with their default values in °C) :

```
tw          = 20.0,
ts          = 20.0,
tde         = 1.0
```

7.4 Boundary forcings for benthic simulations

The STANDALONE benthic model can be run without any pelagic input or by providing a constant deposition flux of organic material and oxygen ventilation rates. Other seasonally-varying deposition rates have to be directly implemented by the user.

The deposition fluxes of organic C and macronutrients are set in the namelist `anforcings_nml` given in Sec. 5.1. These rates are added to the pelagic bottom concentration of organic detritus and oxygen with a simple Euler forward integration.

The default values are

```
botdep_c    = 1.0
botdep_n    = 0.015
botdep_p    = 0.00095
```

```
botdep_si    = 0.015
botox_o      = 0.1
```

8. Test cases

We present here two pelagic test cases and one benthic example obtained with different biological settings.

The pelagic tests are computed at the Equator where light is constant throughout the year (250 W m^-2) with a smooth sinusoidal function between night and day. The simulation duration is of one year and daily averages are computed.

The test cases presented are obtained by varying some of the parameters in the namelists (`standalone.nml`, `bfm.nml` and `Param.nml`). Namelists are renamed for each of the tests (ex: `standalone_test_#.nml`, where # is the test number). To perform the desired test one needs to link the appropriate namelist to the original namelist.

```
> ln -s standalone_test_#.nml standalone.nml
```

In test 1, a standard setting with the inclusion of all phytoplankton groups (diatoms, nanoflagellates, picophytoplankton and dinoflagellates) is presented (fig. 1), while in test 2 only the first three groups are considered (fig. 2).

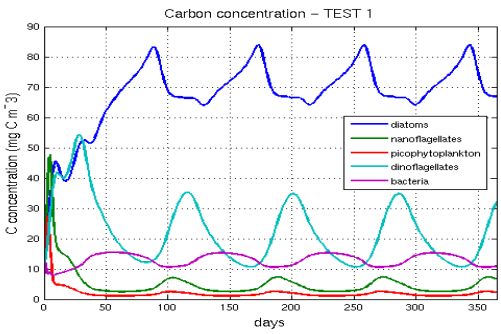

Figure 1: Bacteria and Phytoplankton carbon (mgC m^-3) TEST 1

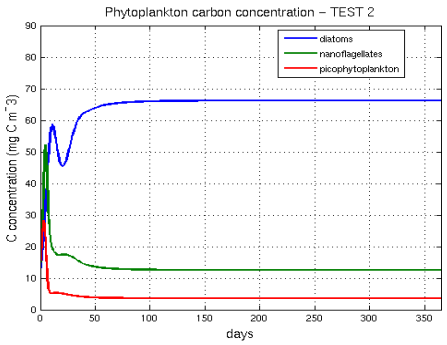

Figure 2: Phytoplankton carbon (mgC m^-3) TEST 2

It is interesting to remark that the addition of dinoflagellates to the system produces an oscillation of phytoplankton biomass with a period of about 90 days, while the system without dinoflagellates stabilizes itself on a steady value.

The benthic example (test 3) uses the full benthic regeneration model described in the BFM Equation Description. The model simulates the adjustment of 30 cm of sediments to constant deposition rates of organic material. The given rates are not sufficient to sustain a full benthic community (apart from aerobic and anaerobic bacteria) but allow the system to reach equilibrium in 10 years of simulation.

Figure 4 shows the results of the penetration depths of oxygen, denitrification layer and organic matter in the sediments.

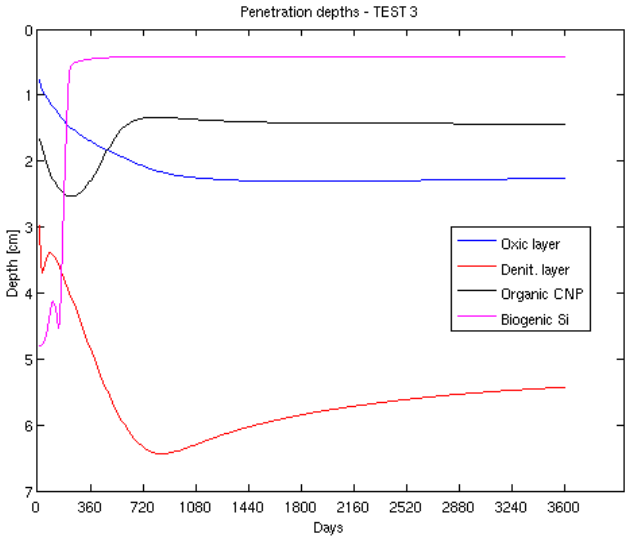

Figure 4: Depths of the oxic and denitrification layers and penetration of organic matter

9. Output visualization

Output values are stored in netCDF according to the CF-1.0 convention (http://www.cgd.ucar.edu/ cms/eaton/cf-metadata/).

The STANDALONE output can be easily visualized with the all-round ncview software (http://meteora.ucsd.edu/~pierce/ncview_home_page.html).

A matlab GUI is currently under construction and will be available soon. For the time being, matlab users can use the `bfm_ncload.m` utility available in the download section of the website.

A. Source directory tree

```
bfm-<version>
├── bin
├── compilers
├── doc
├── include
├── lib
│   ├── IFORT
│   └── NECSX6
├── modules
│   ├── IFORT
│   └── NECSX6
└── src
    ├── BFM
    │   ├── Ben
    │   ├── Bennut
    │   ├── Forcing
    │   ├── General
    │   ├── Light
    │   ├── Oxygen
    │   ├── PelB
    │   ├── PelBen
    │   ├── include
    │   ├── proto
    │   └── scripts
    ├── nml
    ├── share
    ├── gotm (not yet implemented)
    ├── pom (not yet implemented)
    ├── nemo (not yet implemented)
    └── standalone
```