BFM user manual

Marcello Vichi (vichi@bo.ingv.it)

Draft version 3.2 - 2011

Contents

1 Introduction

1.1 Synopsis

This manual describes the major technical features of the Biogeochemical Flux Model (BFM version 4) and provides information to understand the code structure. It also provides examples of the possible diagnostics that can be activated and illustrates the advanced features that allow the user to modify the model adding new state variables and components. The BFM is a direct descendant of the European Regional Seas Ecosystem Model (ERSEM) and shares most of its characteristics with the original formulations (Baretta et al., 1995; Baretta-Bekker et al., 1997). The major difference is that BFM focuses on the biogeochemistry of marine ecosystems, which is where it takes its name.

The philosophy of the BFM and the basic biogeochemical processes which can be simulated are described in Vichi et al. (2007b) and partly in a forthcoming separated document. The users familiar with ERSEM will find that most of the notation described by Blackford and Radford (1995) has not changed and that the state variables are indicated with the same names. Additional variables have been named accordingly. The structure of the code was instead extensively modified with respect to the ERSEM-II code. This was partly due to the F90 coding style and partly because this version of the BFM directly originates from ERSEM-III, which was coded within the OpenSESAME environment. OpenSESAME is a natural successor of the SESAME software (Ruardij et al., 1995) developed at NIOZ during the last phase of the ERSEM-II project.

The major reason for a full rewriting of the ERSEM structure lied in the growing need to couple biogeochemical processes with hydrodynamical model of various shapes. The necessity to have a flexible system which can be embedded in the existing state-of-the-art ocean general circulation models (OGCM) required a re-organization of the structure, yet the fundamentals of the coupling strategy are very similar to the first coupled implementations.

1.2 Couplings with hydrodynamical models

The BFM is now being coupled with different ocean models and some of these couplings are described in Part II of this manual. The idea is that the BFM is the computational core for local biogeochemical processes and as such it has no description of transport processes. A good candidate for coupling with the BFM is thus any physical model with modular or object-oriented routines for the transport of tracers. This is generally possible because temperature and salinity are tracers and their transport dynamics are usually numerically solved by means of the same discretized methods. The same transport routines can be applied to the biogeochemical state variables.

We will not discuss the numerics of the coupling between ocean transport processes and biological source/sink rates. The question will be addressed mostly from a technical viewpoint, by for examples taking into account the procedures by which the memory can be shared with an OGCM that already includes passive tracers or how the additional BFM memory can be made available to a model without a structure for biological state variables.

To simplify the description, the BFM will be described in the STANDALONE implementation, which is essentially a box model with a given depth where pelagic processes are homogeneous, and a bottom layer where the benthic processes occur driven by the settling of pelagic organic

matter. No vertical or horizontal processes are considered and the system is decribed by a set of discretized ordinary differential equations.

The quick-start guide presented in the next section provides all the technical information to set up and run test simulations with the STANDALONE version. The other testing ground of the BFM with multidimensional physical models are with the Princeton Ocean Model, the NEMO model (URL) and the Generalized Ocean Turbulence Model (GOTM, `http://www.gotm.net`), .

1.3 Acknowledgments

The core of the BFM system software was developed by Piet Ruardij (NIOZ) and Marcello Vichi (INGV-CMCC). Additional software developments have been provided by Momme Butenschoen (STANDALONE model), Lavinia Patara (iron and carbonate chemistry), Marco Zavatarelli and Luca Polimene (phytoplankton and bacteria). The coupling with NEMO was done by M. Vichi with the collaboration of G. Mattia. The coupling with POM was done by M. Zavatarelli, M. Butenschoen, E. Clementi and M. Vichi. The coupling with GOTM was done by M. Vichi, P. Ruardij and K. Bolding. The scientific support have been provided by Job Baretta, Nadia Pinardi, Hanneke Baretta-Bekker, Marco Zavatarelli.

2 Quick-start guide

The default basic configuration of BFM is the STANDALONE model, which simulates the biogeochemistry of a water volume with given depth. 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 prescribed or 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.

2.1 Installation

After downloading the tarball from the website, create a directory for the BFM and cd into that directory. Uncompress and extract the contents of the tarball. The downloaded file will have a different name based on the version. For example:

```
% mkdir $HOME/BFM
% cd $HOME/BFM
% 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 default test case simulates the seasonal cycle of pelagic biogeochemistry in a temperate sea (`bfm-run/standalone/temperate.pelagic`).

The full directory tree of the BFM system is given in Fig. 3.1.

2.2 System Requirements

The currently supported architectures are

- linux
- Mac OSX (Darwin)
- AIX

The software requirements are:

- TCL shell (version 8.4) `tclsh`, used automatically during compilation time to generate the code from the model template (see Sec. 4.2).
- FORTRAN 90/95 compiler. Under linux and Mac OSX the model can be currently compiled with gfortran. For AIX xlf90 is required.
- NetCDF library (http://www.unidata.ucar.edu/software/netcdf) . It is mandatory that the library has been compiled with the same compiler used for the model compilation since the F90 netcdf module is used.

2.3 Compilation

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

BFMDIR The root directory of the BFM model (the path to the installed bfm-<version> directory).

The `Makefile` is automatically generated from a configuration script in the `build` directory by means of the `mkmf` Perl script (`http://www.gfdl.noaa.gov/~vb/mkmf.html`) found in the `bin` directory. The configuration script `build/config_STANDALONE.sh` contains a user-dependent part with the compilation options and the name of the executable that will be generated. The options are set by adjusting (or adding) an architecture file in directory `$BFMDIR/compilers`. Default file is `xlf90.inc`. The standard GNU `gmake` variables are used for compiler and archiver names. Remember to add the right path for the NetCDF library files in the appropriate .inc file.

```
cd $BFMDIR/build
./config_STANDALONE.sh
cd BLD_STANDALONE
gmake
```

This will create a directory `BLD_STANDALONE` with the `Makefile`. The compilation is done in the usual way by launching the GNU `gmake` command. This will build an executable binary file called `bfm_standalone.x` in `bin`. Additional directives are included:

`gmake neat` will remove object files only;
`gmake clean` will remove the executable and module files.

2.4 Running

The standard executable has to be located in the same directory of the namelist files. A symbolic link is already provided in directory of the standard test case (see Sec. 2.7) `$BFMDIR/bfm-run/standalone/te`
The model is run by executing `./bfm_standalone.x` or `./bfm_standalone.x &> outputfile` to redirect the output messages to a file in bash. Most of the BFM settings can be controlled via namelists. The STANDALONE model behaviour is controlled with 3 major namelist files in the `temperate.pelagic` directory.

2.4.1 `standalone.nml`

This file contains the namelist `standalone_nml` (Table 2.2) which sets the dimensions of the model, the numerical integration details and the namelist `time_nml` controls the time manager (Table 2.4, an extension of the one developed by the GOTM team under the GPL license, `http://www.gotm.net`). The namelist `forcings_nml` controls the type of forcing functions (analytical or from file, see Sec. 2.6) and the presence of specific user data. This last option is totally user-defined and it is meant to provide a facility for adding events to the simulation, such as the inoculation of food at a certain time or to specify a chemostat experiment. The template routine `src/standalone/external_data.F90` is given as an example of the way the time interpolation may be handled.

2.4.2 `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 given in Table **??**.

The namelists `bfm_init_nml`, `bfm_ben_init_nml` give the initial values to the state variables (identified with a trailing 0 after the name of the state variable) for the model domain. For example, in the pelagic system the values are given as:

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

or in the benthic model as:

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

This file also contain the namelist for the storage of variables and other diagnostics, as completely detailed in Chapter 5.

2.4.3 `Param.nml`

This is the core of the BFM, where most of the available options can be set. A complete description of the available options for changing the code structure is given in Section 3.4. We report in Table 2.10 only the major parameters of interest for the default test case (Sec. 2.7).

2.5 Numerical integration

Three integration schemes are implemented in the STANDALONE model. They 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, indicating with $S(c_n)$ the right-hand-side source term computed with concentration c_n, they are written as:

- the Euler scheme (`method=1`):
  $c_{n+1} = c_n + S(c_n)\Delta t$
- the Runge-Kutta scheme of 2$^{\text{nd}}$ 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: $c_n = c_n + \gamma(c_{n-1} - c_n + c_{n+1})$

2.6 Forcing functions

The STANDALONE model can be forced with different kind of forcing functions to provide the physical information for the aquatic system (e.g. temperature, irradiance, etc.). Two options are currently available: analytical forcing functions and data from an external file. All external data formats and time interpolations are derived from the code provided by GOTM under the GPL.

2.6.1 Analytical forcing functions

This option is activated by setting forcing_method=1 in the namelist The simple analytical forcing functions 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 with the parameters in Table 2.6 which can be changed in the namelist file `standalone.nml`. The basic concept behind all forcing functions is the interpolation of some (northern hemisphere) summer and winter values by a sine wave, considering them as the two extreme values. This simple case is used for salinity forcing data (`sw` and `ss` in Table 2.6),

while for temperature, in addition to the sine wave, it is possible to superimpose a daily excursion (constant all over the year).

Light forcing is interpolated by a sine wave from a winter extreme value and a summer extreme value (`lw` and `sw` in Table 2.6) to obtain the actual value of average daylight for each day of the year. These variables are as well intended as the placeholders for instantaneous light values and average daylight, i. e. the mean value of light energy over the light period:

$$\frac{1}{daylength}\int_{\tau_{dawn}}^{\tau_{dusk}} light(t)dt$$

The daylength is computed by the model as a function of latitude. 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/24`. 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 daily period that is available to the water box is the same.

2.6.2 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 forcings_nml given in Table . These rates are added to the pelagic bottom concentration of organic detritus and oxygen with a simple Euler forward integration.

2.6.3 Environmental data from file

This option allows the user to input the environmental data directly from a file. The required input data are seawater temperature (variable `ETW`, ^oC), salinity (`ESW`), irradiance (in W m^{-2}, converted directly into the model variable `EIR`) and wind velocity (`EWIND`, m s^{-1}). Any other parameter can be included by modifying the file `standalone/external_forcing.F90`. The data file format is equivalent to the one of GOTM, an ASCII file with a first column indicating the date (`yyyy-mm-dd hh:mm:ss`) and one column for each parameter separated by space(s).

2.7 Test cases

The standard test case is found in directory `bfm-run/standalone/temperate.pelagic`. It simulates the seasonal cycle of a well-mixed temperate coastal sea, with mean depth 5 m. This example is to considered as a template for other application, as it is not meant to describe any real-world application. The simulation starts in January 2000 and terminates in 2010 with a maximum time step of 8640 s in the adaptive Runge-Kutta solver and with the output stored every 10 days. After 2010 the system starts showing more chaotic fluctuations typical of this kind of non-linear systems. The time evolutions of temperature, irradiance, nutrients (the system is sustained with regenerated nutrients only), chlorophyll and secondary gross production are provided in Figs. 2.1-2.3 for comparison.

2.8 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 visualized with the all-round `ncview` software (`http://meteora.ucsd.edu/~pierce/ncview_home_page.html`) or with `ncbrowse`. Matlab users can use the `bfm_ncload.m` utility available in the script directory (a working netcdf toolbox installation is required).

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

Table 2.2: Main parameters in the standalone.nml file

name	*kind*	*description*
timefmt	integer	Format of the time limits and loop: 1. MaxN only. Give number of iterations, fake start time is used. 2. Start and stop dates. MaxN calculated. 3. Start and MaxN. Stop time calculated. 4. 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

Table 2.4: Main parameters in the time_nml namelist

name	*kind*	*description*
forcing_method	integer	Choice of the external forcing functions 1. analytical forcings 2. from file 3. interactive fluxes (not yet implemented)
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
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})
forcing_file	char	filename for external forcings (forcing_method = 2)
use_external_data	logical	read external data (user defined)
data_file	char	filename for external data

Table 2.6: Parameters for the external forcing functions

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

Table 2.8: Parameters that control the STANDALONE setup and output file (standalone.nml)

name	*kind*	*description*
CalcBenthicFlag	integer	Switch for the benthic model 1. no benthic model; 2. simple benthic return; 3. benthic organisms and intermediate complexity nutrient regeneration; 4. 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 for Z3, Z=2 for Z4)
CalcMicroZooplankton(Z)	logical	Switch for microzooplankton (Z=1 for Z5, Z=2 for 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: the variable for chl content P1l is used to store otimal irradiance in Wm-2 2. variable chl:C ratio (note: P1l is in mg chl)

Table 2.10: Parameters that control model formulation in Param.nml

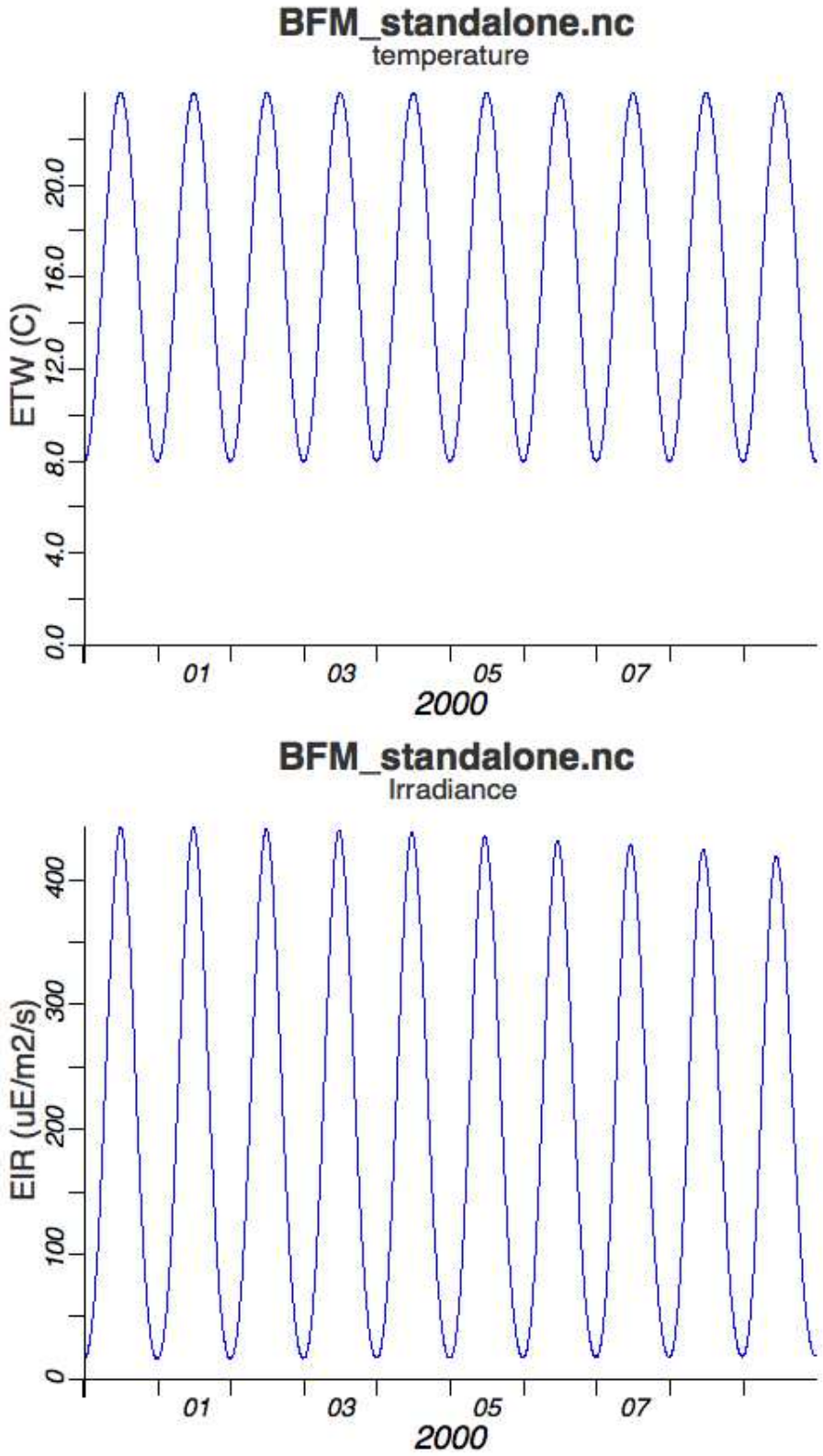

Figure 2.1: Sample output of the STANDALONE test case temperate.pelagic. a) temperature; b) irradiance

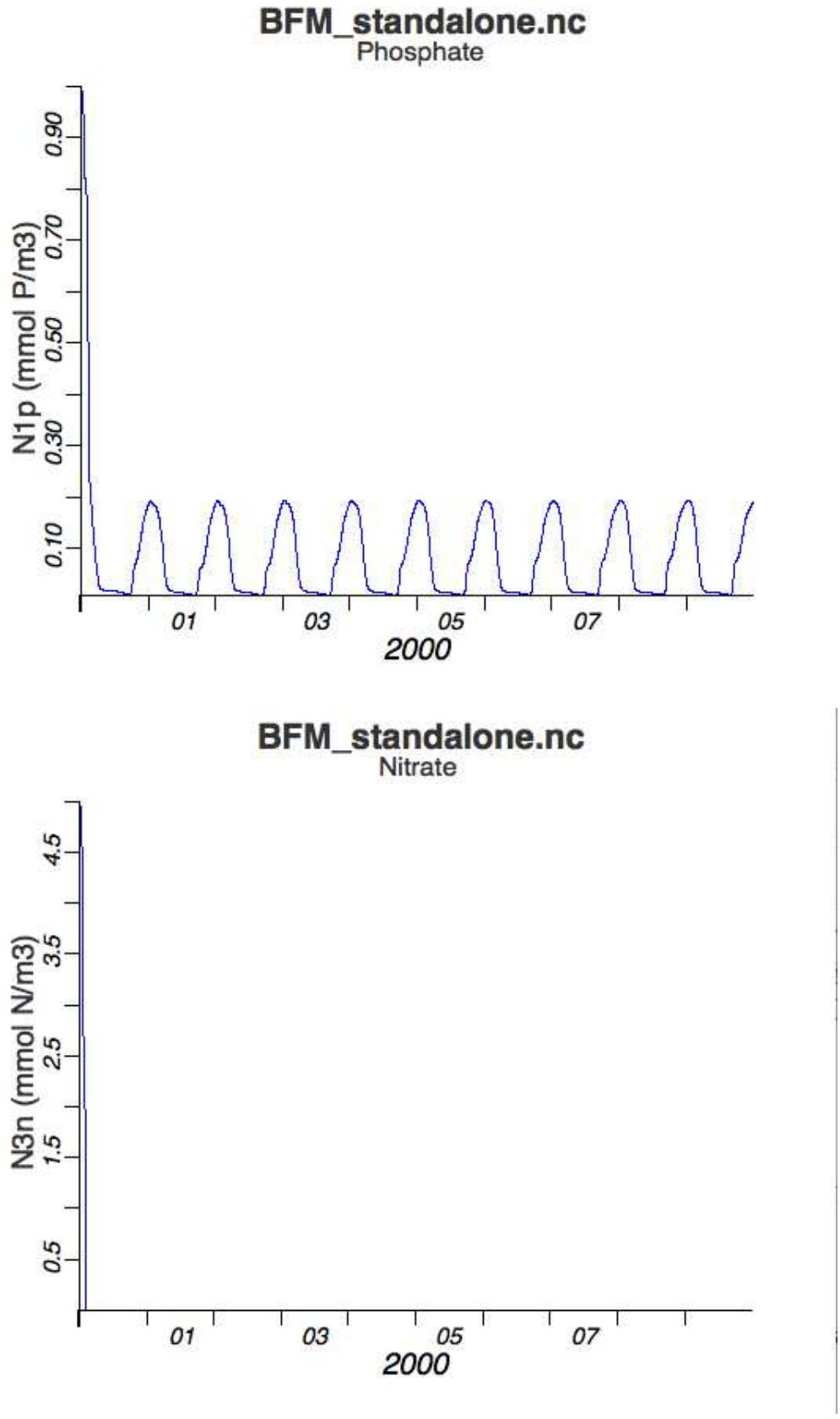

Figure 2.2: Sample output of the STANDALONE test case temperate.pelagic. a) phosphate; b) nitrate

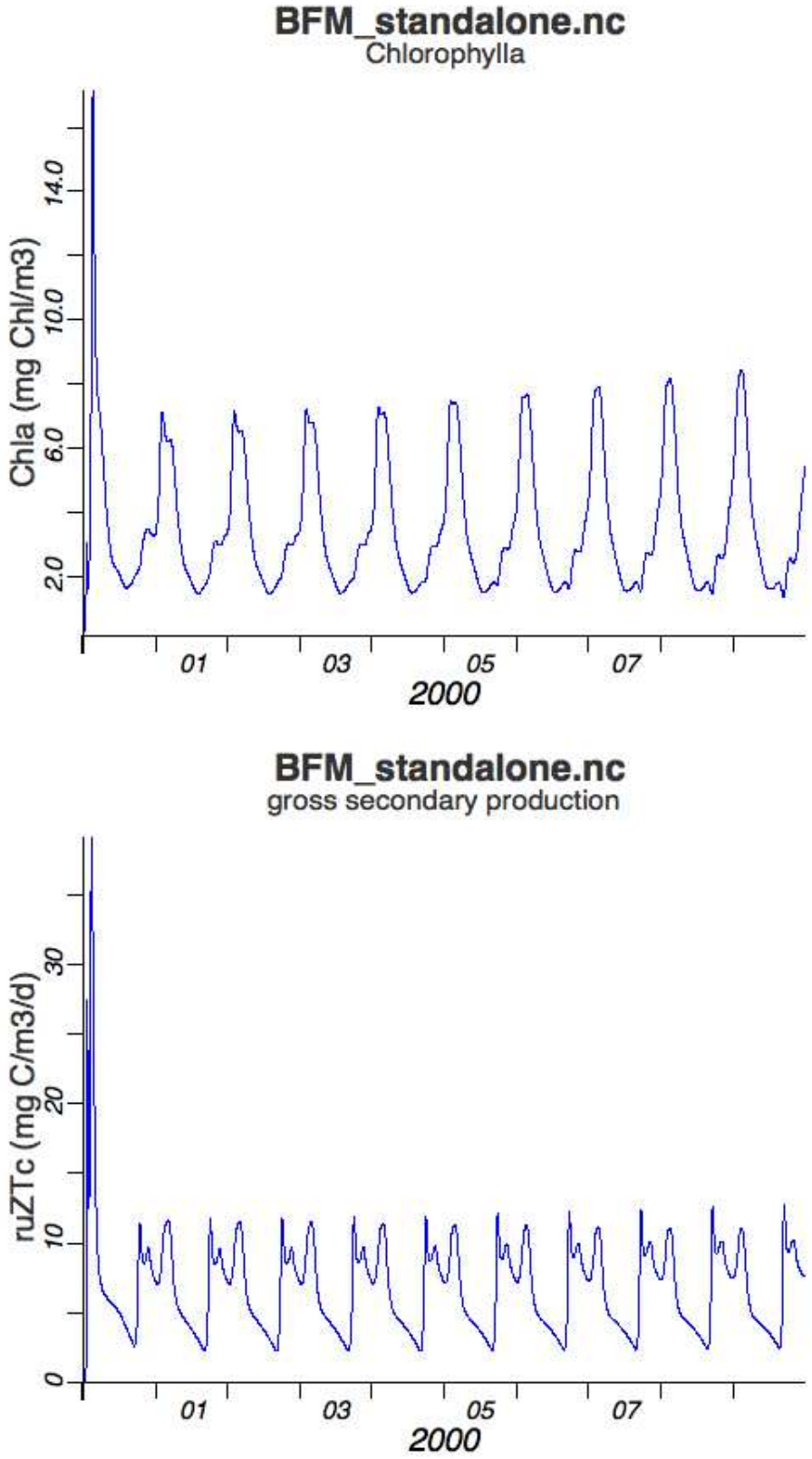

Figure 2.3: Sample output of the STANDALONE test case temperate.pelagic. a) Chlorophyll-a; b) gross secondary production

3 Structure of the code

3.1 Coding rules

The BFM is written in F90 and it can generally be compiled with any F90 compiler. The development process is done with the publicly available Intel Fortran Compiler (ifort) under linux platforms, with a special branch of development on NEC SX-series vector machines. Any other compiler or platform is not directly supported, although some help will be provided for smooth integration with the existing line of development.

There are no fixed coding rules apart from standard clean programming. For historical reasons, the subroutine and symbols of the BFM biogeochemistry core are indicated with C capitalized long-names (e.g. EcologyDynamics), even if F90 does not distinguish capital letters. Constants and main memory arrays are generally indicated with capital names, to highlight their global static behavior. It is recommended to follow this style when possible.

3.2 Directory tree

BFM is spread over the classical directory tree (`bin/, lib/, src/, include/`), which is populated when the Standalone model is compiled. For all the other couplings the user should refer to the specific chapter in Part II. The content of the `src/` directory and a description of the subdirectories is depicted in Fig. 3.1. The routines for the biogeochemical processes are contained in the BFM dir and

3.3 Memory layout

The BFM is structured similarly as the original ERSEM used to be. Pelagic variables are defined as virtually 3-dimensional, while benthic variables are 2-dimensional. Only the sea-points are stored in memory, and thus the BFM has no knowledge of any land-sea mask or spatial relationships between grid-points in general. The same applies for the benthic memory, which is defined only where a benthic system underlies a water column. The number of bottom points is the same as the number of surface points, therefore the 2-D arrays can also be used for the surface variables and/or surface fluxes. These definitions are also maintained when the model is applied to a 1-D vertical setup. In that case, the pelagic variables have the spatial dimension of the number of vertical layers while the benthic and surface variables have dimension 1 (the bottom and surface levels, with the index depending on the definitions of the physical model). In a 0-D setup (as in the STANDALONE test case described in Sec. 2.7), the same grid-point is at the same time a bottom, surface and ocean layer.

Fig. 3.2 illustrates the memory layout of pelagic variables. They are gathered in a single 2-dimensional array holding the number of different variables and the number of ocean grid-points:

```
D3STATE(NO_D3_BOX_STATES,NO_BOXES)
```

and the same is done for the benthic variables:

```
D2STATE(NO_D2_BOX_STATES,NO_BOXES_XY).
```

```
src
|------- BFM
|          |-- Ben  (benthic organisms)
|          |-- Bennut  (benthic nutrient rigeneration)
|          |-- Forcing  (helper routines)
|          |-- General  (BFM structure definition
|          | and memory allocation)
|          |-- Light  (light availability and utilization)
|          |-- Oxygen  (oxygen dynamics and reaeration)
|          |-- CO2  (DIC dynamics and reaeration)
|          |-- PelB  (pelagic organisms and biochemistry)
|          |-- PelBen  (pelagic-benthic coupling)
|          |-- include  (macro definitions)
|          |-- proto  (file templates for model structure)
|          `-- scripts  (tcl script to generate model structure)
|
|-- getm  (files for the BFM-GETM coupling)
|-- gotm  (files for the BFM-GOTM coupling)
|-- nemo (files for the BFM-NEMO coupling)
|-- pom (files for the BFM-POM coupling)
|-- share
(files shared by all configurations and storage routines)
`-- standalone (files for the Standalone configuration)
```

Figure 3.1: Source directory tree and description of the directory contents.

Each variable memory is accessed by means of a pointer to the corresponding raw containing all the grid-points value. This pointer array has the same naming convention of ERSEM. The rows are numbered with named constants starting with the letters pp (ppP1c for phytoplankton carbon, ppN1n for nitrate or ppR1p for dissolved organic phosphorus, etc.). When the BFM is coupled to a 3-D model, then a mapping function must be taken into account as shown in Fig. 3.2 in order to compute the transport terms.

In addition to this direct mechanism, the memory can be accessed by means of *group functions*. These functions have been created to allow the addition of any number of sub-groups in the same trophic compartment, such as diatoms, nanoflagellates, picoplankton and dinoflagellates in the group of phytoplankton. The group function is in this case called `PhytoPlankton(no_of_subgroups, no_of_components)`, and allows to access the memory `P1c` by writing

```
PhytoPlankton(iiP1,iiC)
```

The components are defined in the model structure file which will be described in the chapter of advanced features. It is now only sufficient to list the constants indicating the constituents, namely `iiC`, `iiN`, `iiP` and `iiSi`. For each defined group there is a named constant starting with the letters `ii` which holds the total number of existing subgroups (e.g. `iiPhytoPlankton`). The existing groups and the syntax to define new others is also given below in Sec. 4.2.

A typical example of the use of groups is to compute the total amount of food available to microzooplankton (in `MicroZooDynamics.F90`):

```
do i = 1 , ( iiPhytoPlankton)
    rumPIc(:, i) = p_suPI(zoo,i)* PhytoPlankton(i,iiC)* &
      PhytoPlankton(i,iiC)/( PhytoPlankton(i,iiC)+ p_minfood(zoo))
    rumc  =   rumc+ rumPIc(:, i)
    rumn  =   rumn+ rumPIc(:, i)* qnPc(i,:)
    rump  =   rump+ rumPIc(:, i)* qpPc(i,:)
end do
```

Compare this code with the equation for food availability and capture efficiency presented in Sec. 2.4.1 of the BFM description and it will show how it is simple to cycle over all the phytoplankton variables, independently of the number and type of the defined subgroups.

The other basic arrays of the memory are the ones containing the time rates of change due to each physiological or ecological interaction. Their structure partly derives from the methodology proposed by Burchard et al. (2005), and partly is inherited from ERSEM. The rate of mass exchange between two state variables is an element of an array `D3SOURCE`, if it is a production term, and `D3SINK` if it is a loss term, both with dimensions `NO_D3_BOX_STATES` × `NO_D3_BOX_STATES` × `NO_BOXES` (Fig. 3.3). It is clear that the two matrices are simmetrical and a predation (negative) rate from variable P1c to Z4c is equivalent to an ingestion rate (positive) of `Z4c` over `P1c`. All the rates are defined positive by convention, such as to obtain the net rate of change for variable of row j, it is sufficient to compute

```
sum(D3SOURCE(j,:,:)-D3SINK(j,:,:))
```

This separation between positive and negative terms is required in order to use positive-definite integration schemes (as when coupling with GOTM, Burchard et al.,).

Fluxes which involve a variable not defined in the model, such as denitrification rates leading to dissolved molecular nitrogen, can be taken into account by using the diagonal of the matrix. The loss of nitrate due to denitrification is therefore a flux from nitrate to nitrate in the sink matrix (`D3SINK(N3n,N3n,:)`, cfr. Fig. 3.3). Similarly, the oxygen production during photosynthesis

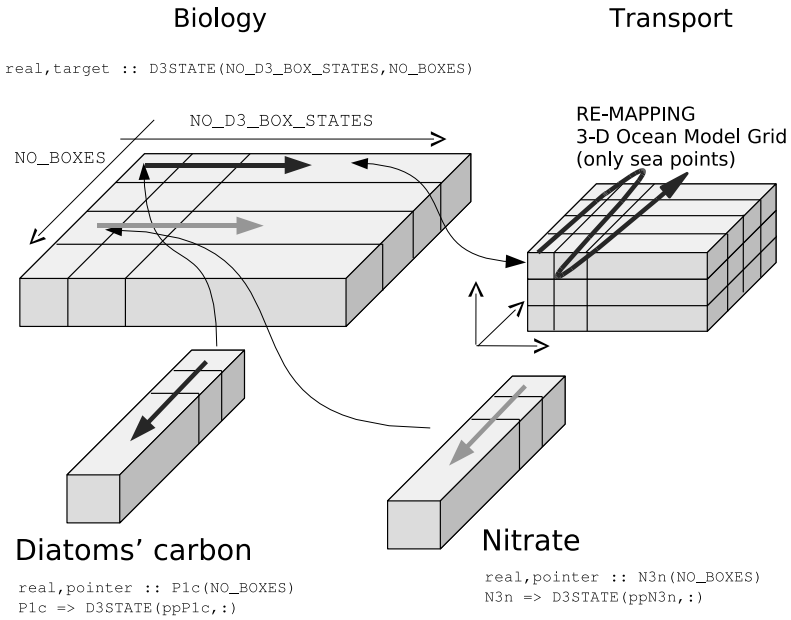

Figure 3.2: Layout of the main memory array for the pelagic variable (D3STATE) and schematic of the remapping into a three-dimensional ocean grid for transport processes.

is a positive flow from `O2o` to itself computed stoichiometrically from the gross carbon production rate.

The same definitions apply for the benthic system, and the arrays are called `D2SOURCE` and `D2SINK` with dimensions `NO_D2_BOX_STATES`×`NO_D2_BOX_STATES`×`NO_BOXES_XY`.

No direct mass exchanges are allowed between pelagic and benthic variables, and therefore the fluxes from the pelagic to the benthic (and viceversa) are seen as diagonal kind of fluxes. Specialized functions have been written to take into account all the possible kind of exchanges and are collected in `General/FluxFunctions.F90`.

3.4 The BFM flow-chart

The BFM core will be described in its Standalone configuration. The main program (Fig. 3.4) consists of 2 subroutines, one for initialization and one for the time-loop. In the coupled configurations, these procuderes are likely to be included in the relative sections of the hydrodynamic model (cfr. the examples in Part II)

3.4.1 Initialization procedures

The initialization of the BFM requires 4 different phases:

1. Initialization of geometric domain (number of boxes/gridpoints). This part depends on the configuration: in the Standalone case, this is done in standalone.F90[init_standalone] while in the other coupled configuration there are specific interface subroutines;

2. Initialization of the set-up via namelist and definition of variable information (`share/api_bfm.F90[init_bfm]` for the Standalone case);

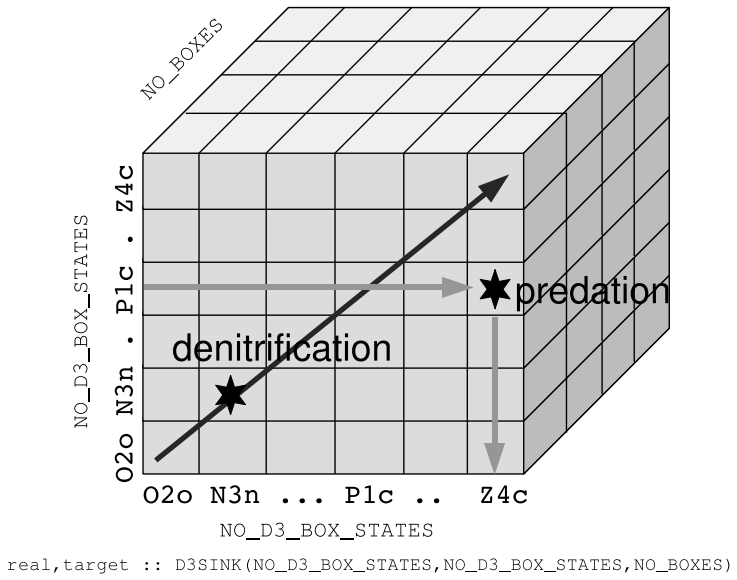

Figure 3.3: Schematic of the D3SINK array collecting the sink terms.

3. Definition of variable information (BFM/General/set_var_info_bfm.F90) and allocation/initialization of pelagic and benthic variables;

4. Initialization of netcdf output.

3.4.2 Time marching

The time marching is driven by the host hydrodynamical model, or in the case of the Standalone model, by a specific built-in routine. The phases of the time stepping shown in Fig. 3.4 are the following:

1. update of the physical forcing functions needed by the biogeochemical model (temperature, irradiance, wind, etc.). This is done by subroutine envforcing_bfm, which in the case of the Standalone model computes the analytical values based on the time of the day/year. When coupled with an ocean model, this routine is the contact point between the two models. Note that information may flow in the two directions because the light attenuation coefficient might also be passed back to the physical model via this routine.

2. computation of the reaction term for the pelagic and benthic models. This is done in EcologyDynamics, which is further explained in the next section.

3. numerical integration of the biogeochemical terms. In the Standalone model there are three different numerical schemes that can be used (see the quickstart guide for the specific namelist values). Note that EcologyDynamics can be called more than one times by this routines when the integration schem is of higher-order.

4. preparation and writing of the NetCDF output, routines calcmean_bfm and save_bfm (the names of the routines can be different for each coupling according to the ocean model structure and naming conventions).

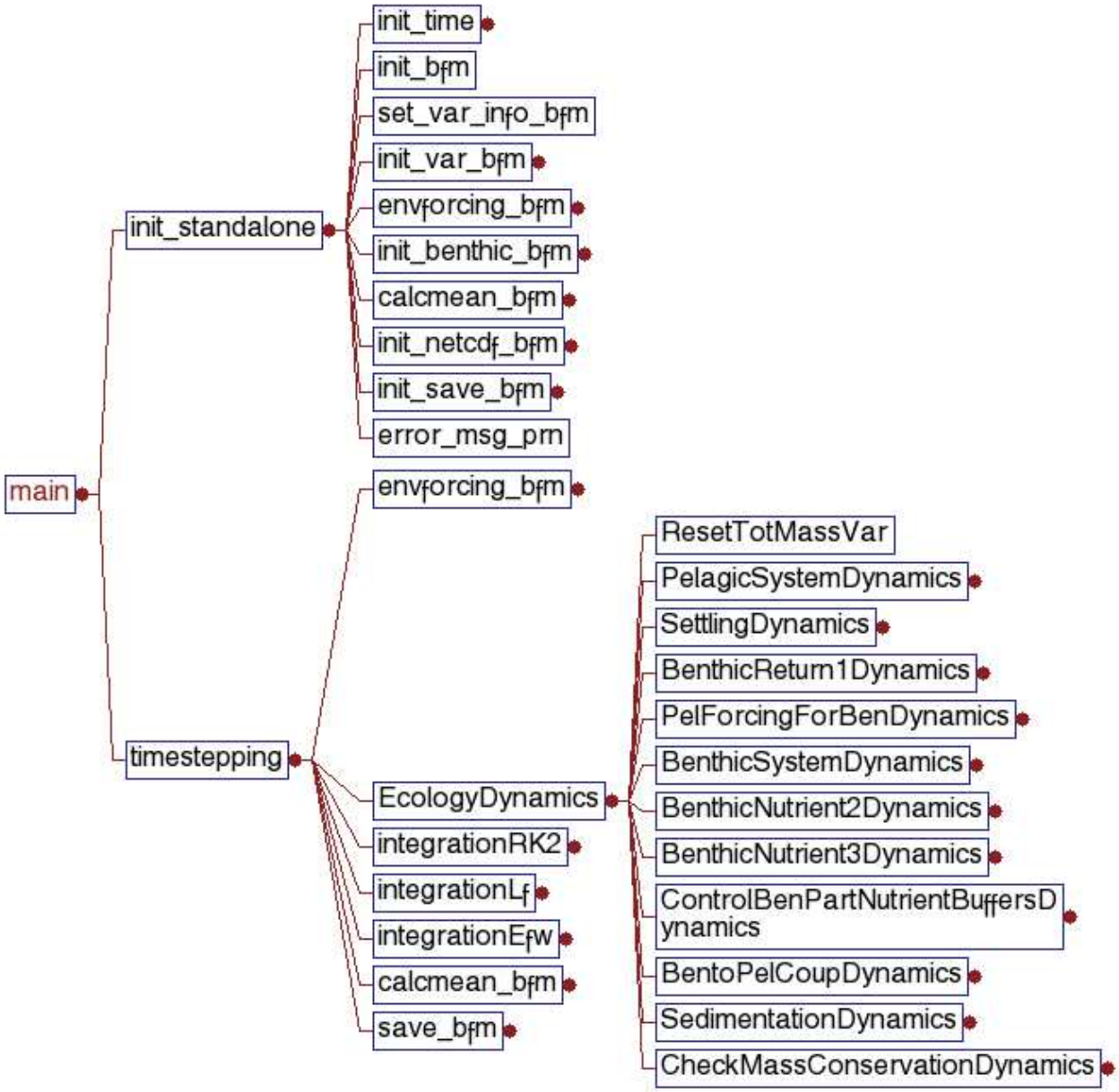

Figure 3.4: Invocation tree of the Standalone main.

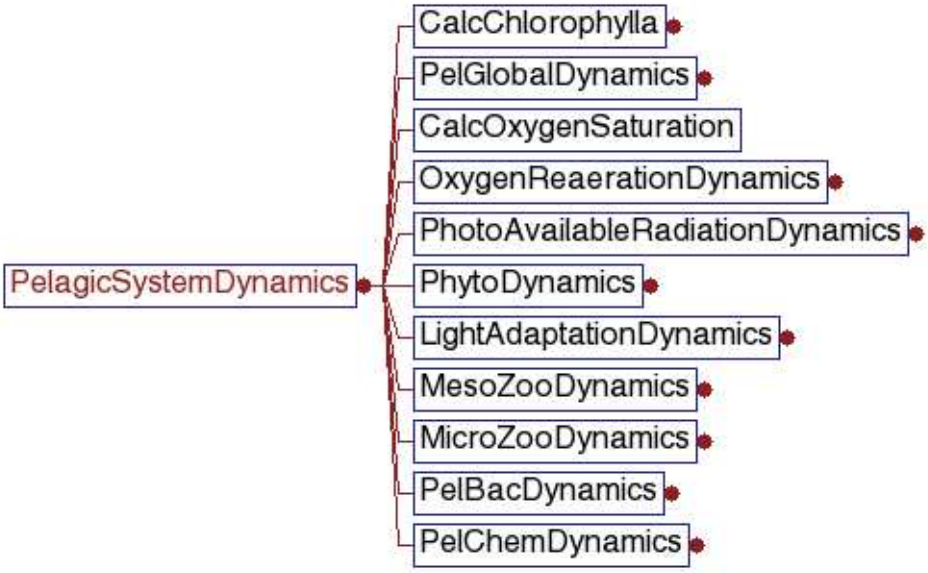

Figure 3.5: Invocation tree for PelagicSystemDynamics

3.4.3 Computation of pelagic reaction terms

The pelagic components are computed in `PelagicSystemDynamics`, which invokes the routines shown in Fig. 3.5. It first sets the diagnostic values for total chlorophyll (sum of all the chlorophyll components in phytoplankton) and for the nutrient:carbon ratios in all the components (`PelGlobalDynamics`) and for oxygen saturation and surface fluxes with the atmosphere.

The routines for the computation of light availability are only used when the parameterization of optimal irradiance is activated (as in ERSEM-II, Ebenhöh et al., 1997), which is done in `Param.nml` by setting `ChlLightFlag=1`.

The standard LFGs, phytoplankton, meso-/ microzooplankton and bacteria are computed in `PhytoDynamics`,`Meso- MicroZooDynamics` and `PelBacDynamics`, respectively. The same routine is modularly used for each sub-group, as in the case of phytoplankton

```
do i =1,iiPhytoPlankton
   if ( CalcPhytoPlankton(i)) then
      call PhytoDynamics( i, ppPhytoPlankton(i,iiC),          &
           ppPhytoPlankton(i,iiN), ppPhytoPlankton(i,iiP),    &
           ppPhytoPlankton(i,iiS), ppPhytoPlankton(i,iiL))
   end if
end do
```

`PhytoDynamics` has the declaration structure depicted in Fig. 3.6. This is an example of how the memory layout presented in Sec. 3.3 is accessed by each submodel of the BFM. To refer to the pointer of each phytoplankton component, it can be used the placeholder function `ppPhytoPlankton(i,j)`, where *i* is the phytoplankton subgroup number (up to 4 in the reference model) and *j* is the component (c, n, p, etc). Using `ppPhytoPlankton(1,iiC)` is equivalent to write `ppP1c`.

The function `flux_vector` is invoked many times by `PhytoDynamics` (Fig. 3.6) because it is used to fill in the source/sink term arrays described in Sec. 3.3. This function is used throughout the BFM modules. In case of a mass exchange term such as the excretion of dissolved carbon we have

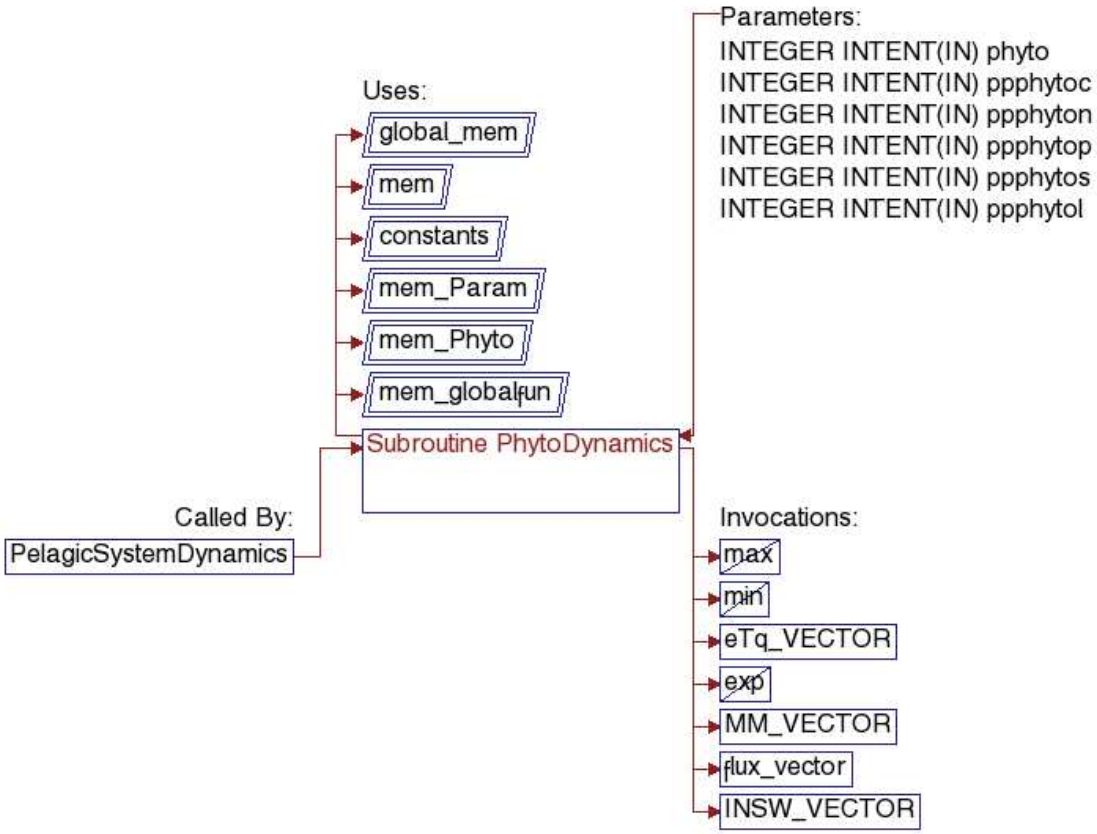

Figure 3.6: Declaration scheme of PhytoDynamics.

```
call flux_vector( iiPel, ppphytoc,ppR1c, rr1c )
```

where the flux is pelagic (`iiPel`), from phytoplankton carbon to dissolved organic carbon (`ppphytoc, ppR1c`) and the value is `rr1c`, which will be assigned both to `D3SINK` and `D3SOURCE` in the appropriate element. The oxygen production rate is instead assigned as

```
call flux_vector( iiPel, ppO2o,ppO2o, rugc/ MW_C )
```

which concerns a pelagic flux (`iiPel`), on the oxygen diagonal (from `ppO2o` to `ppO2o`) and it `MW_C` is a stoichiometric conversion of the gross primary production (`rugc`). Since the last term is positive, this flux will be assigned to `D3SOURCE`.

The same methodology is applied for all the other pelagic fluxes and also in the benthic system. After the computation of all the reaction terms for pelagic organisms, there is the call to pelagic chemistry (`PelChemDynamics`, see Fig. 3.5). This part computes all the parameterized bacteria-mediated biochemical processes such as nitrification and denitrification, reoxidation of reduction equivalents and dissolution of biogenic silica.

3.4.4 Computation of benthic reaction terms

The BFM has 3 different benthic models:

1. a simple nutrient regeneration model (a.k.a. benthic return);
2. a benthic return with benthic organism dynamics;
3. a dynamical layer model owith full nutrient regeneration and benthic organism dynamics.

The models are selected by setting the corresponding flags in the `Param.nml` namelist file which activate the following subroutines (cfr. Fig. 3.7):

`BenthicFlag>0`: any benthic model activates the settling of variables on the sediments and the benthic-pelagic coupling functions in the `PelBen` directory:
`SettlingDynamics`,
`ControlBenPartNutrientBuffersDynamics`,
`BentoPelCoupDynamics` and
`SedimentationDynamics`;

`BenthicFlag=1`: the simple benthic return is implemented in `BenthicReturn1Dynamics`;

`BenthicFlag=2`: this flag activates the benthic organism model (`BenthicSystemDynamics`) and thus a modified benthic return model is used (`BenthicNutrient2Dynamics`) which requires additional oxygen dynamics (`BenOxygenDynamics`);

`BenthicFlag=3`: the full benthic model activates both the `BenthicSystemDynamics` and the regeneration model with all the nutrients (`BenthicNutrient3Dynamics`).

Benthic organisms are computed as in the pelagic system within BenthicSystemDynamics. Firstly, the different bioturbation rates are computed considering the current benthic biomass distribution (Fig. 3.7). Secondly, all the benthic organisms are called sequentially, and separately, filter feeders and benthic bacteria (both aerobic and anaerobic). Memory addresses for the carbon and nutrient components are accessed by means of unique integer constants as for the pelagic variables:

```
if ( CalcBenOrganisms(iiY1)) then
  call BenOrganismDynamics( iiY1, ppY1c, ppY1n, ppY1p)
end if
if ( CalcBenOrganisms(iiY2)) then
  call BenOrganismDynamics( iiY2, ppY2c, ppY2n, ppY2p)
end if
if ( CalcBenOrganisms(iiY4)) then
  call BenOrganismDynamics( iiY4, ppY4c, ppY4n, ppY4p)
end if
if ( CalcBenOrganisms(iiY5)) then
  call BenOrganismDynamics( iiY5, ppY5c, ppY5n, ppY5p)
end if
if ( CalcBenOrganisms(iiY3)) then
  call FilterFeederDynamics
end if
if ( CalcBenBacteria(iiH1)) then
  call BenBacDynamics( iiH1, ppH1c, ppH1n, ppH1p)
end if
if ( CalcBenBacteria(iiH2)) then
  call BenBacDynamics( iiH2, ppH2c, ppH2n, ppH2p)
end if
```

Filter feeders are treated separately from the other organisms because of their relationship with the water column.

The different nutrient models require additional subroutines. Particularly, the full nutrient model calls a specialized subroutine for each nutrient equation (`BenAmmoniumDynamics`,`BenNitrateDynamics`, etc .). It is important to remember that the first two benthic models do not resolve the vertical distribution of the state variables in the sediments and therfore there is no activation of the layer depth variables and shifting dynamics (`BenDenitriDepthDynamics` and `BenNitrogenShiftingDynamics`).

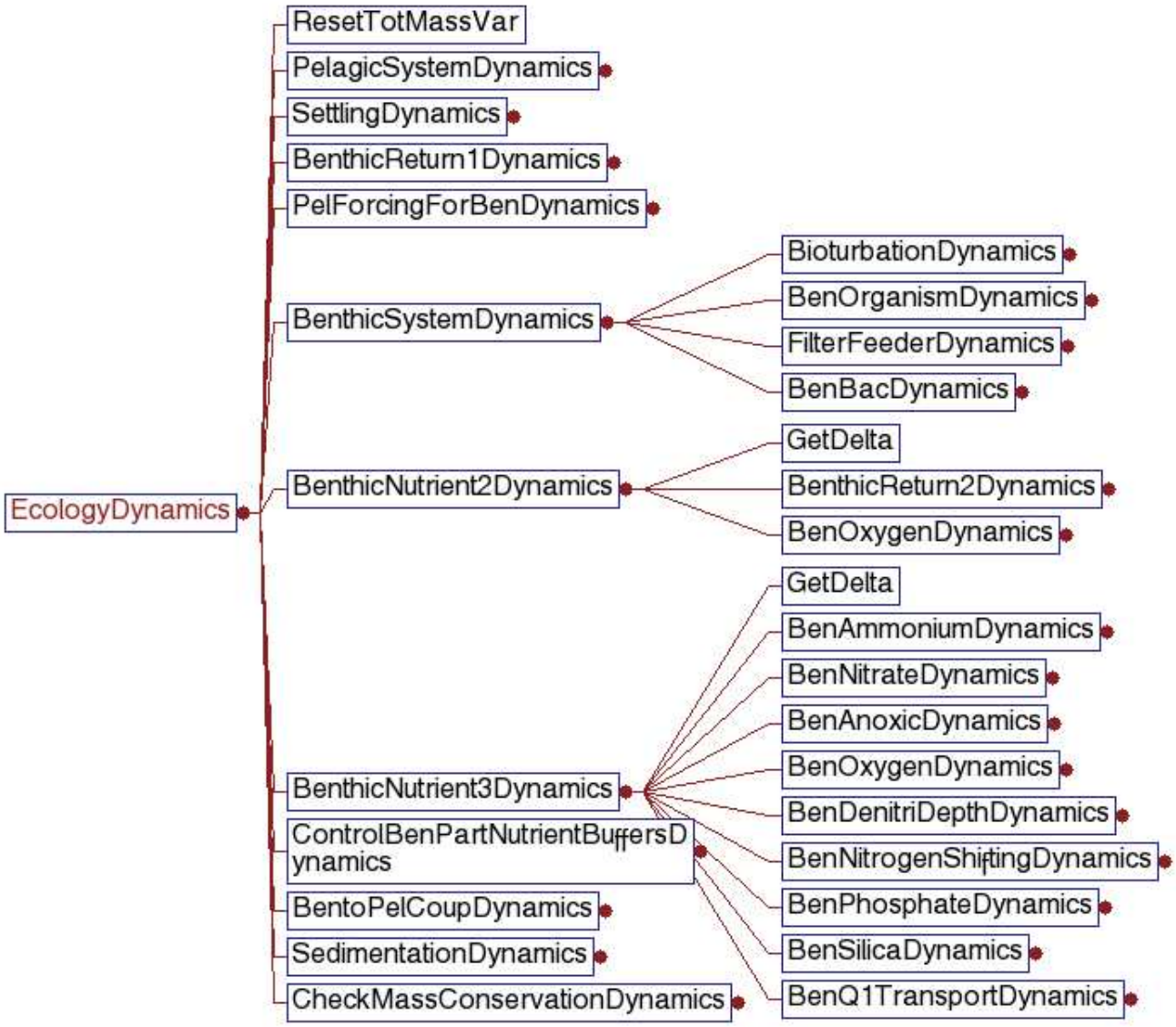

Figure 3.7: Invocation of benthic routines

4 Advanced features

The BFM is conceived in a modular way and the number of state variables and biogeochemical basic components can be modified according to the modelling needs. This section addresses the advanced features that are useful to change the structure of the model and to opitmize some beahior at compilation.

4.1 Pre-compilation macros

There are 2 major macros that belong to the core BFM source code. Both macros are "negative", which means that their options are activated by default and needs to be deactivated when desired.

NOT_STANDALONE: This macro controls the main memory structure of the BFM. In default configuration the BFM allocates its own memory array. When the macro is defined at compilation (`-DNOT_STANDALONE`) all the variables belonging to the main memory are defined as pointers and are not allocated during the initialization phase. This is the configuration used in combination with hydrodynamical models that already defines a memory array for biogeochemical variables (e.g. GOTM, see Part II).

BFM_NOPOINTERS: by default, all the F90 symbols of the BFM state variables are defined as pointers to the main memory (cfr. Sec. 3.3). By defining the macro `-DBFM_NOPOINTERS`, it is possible to substitute all the pointers with the actual memory location. For instance, all the occurrences of the keyword `P1c(:)` or `H1c(:)`, which are pointers to the memory address of phytoplankton and benthic bacteria carbon contents, are substituted with the name `D3STATE(ppP1c,:)` and `D2STATE(ppH1c,:)`, respectively. To facilitate this feature, it is requested that all the references to the pointers in the code are written with the indication of the number of dimensions: `P1c` should always be indicated as `P1c(:)`, otherwise a precompiler error is generated when `BFM_NOPOINTERS` is defined. (Notice that this option is the default when using the NEC-SX architecture).

4.2 Model structure changes

As briefly described in the Introduction, BFM is built in a modular and flexible way, because each state variable is not defined directly in the code but through a model template (`General/GlobalDefsBFM.model`). All the ancillary F90 memory layout files (`AllocateMem.F90, INCLUDE.h, ModuleMem.f90, set_var_info_bfm.f90`) are automatically generated from this meta-data structure by way of a script written in `tcl` (`script/GenerateGlobalBFMF90Code`). The prototypes of these files are found in directory `proto` and filled in with the user-defined state variables and diagnostics.

The structure of the model template is divided in sections with specific headings. Sections can be repeated many times in the file and there is no fixed sequence. It is however suggested to follow a conceptual construction of the model, by grouping all the state variables and diagnostics as done in the standard released structure. Other examples of different model structures are available in the `General/Configurations` directory (the file with the extension `.standard` is the one found in the `General` directory).

3d-state: this section is used for all the state variables, which are defined as being virtually three-dimensional. This means that they exist in the whole domain, be it 0-, 1- or 3-dimensional. An example of state variable definition is shown in Box 4.1.
According to the theoretical model description, variables are defined as chemical functional families (CFF) and identified by their main biochemical components, which are indicated according to the classical ERSEM syntax (Blackford and Radford, 1995). State variable definitions require 3 pieces of information, separated by colons: code symbol with components, long name and units. Components are indicated by one single letter and currently the following are reserved: c=carbon, n=nitrogen, p=phosphorus, s=silica, l=chlorophyll, f=iron. Any other new component (either atomical or molecular) can be added, provided that the user include the relative dynamics in an appropriate piece of code. The syntax for a multi-component variable is `varname[cnp...]`, and it is also possible to define group of state variables, such as PhytoPlankton or PelDetritus as shown in Box 4.1. This feature gives the advantage to cycle over all the group components with one single index (see the example in Sec. 3.3).

2d-state: all the features described above are also valid for the state variables belonging to the benthic 2-D system. An example of benthic variable definitions is given in Box 4.2.

1d-variable: this section defines the variable without a spatial component (more appropriately defined as 0-D variables). These are only function of time and defined as scalars.

2d-variable: this section includes the surface variables, originally identified as benthic variables. However, this section is also used to define surface variables such as wind velocity. In a 1D coupled model, this is a scalar, while it is a one-dimensional array holding all the surface (bottom) points in a coupled 3D model.

3d-variables: this section is used for all the ancillary diagnostic variables defined in the whole domain. Typically, this section is used for temperature, salinity, and other environmental or diagnostic variables. Environmental variables obtained from external data files or hydrodynamical models are indicated with capital symbols starting with E (e.g. ETW = environmental temperature of water).

The configuration can be also modified at compilation time by means of a set of macros. The statement `-if-exist` found after one of the above declarations indicates that the following set of variables will be defined when the macro is included in the compilation:

```
3d-variable -if-exist INCLUDE_PELCO2
    DIC         : Dissolved Inorganic Carbon : umol/kg
    CO2         : CO2(aq) : umol/kg
    pCO2        : Oceanic pCO2 : uatm
    HCO3        : Bicarbonate :  umol/kg
    CO3         : Carbonate : umol/kg
    Ac          : Alkalinity : umol eq/kg
    pH          : pH : -
end
```

This is usually done by changing the values of certain logical flags in src/Rules.make that adds the `-DINCLUDE_PELCO2` statement to the compilation command. The currently included macros are the following:

INCLUDE_PELCO2 Defines the pelagic variables of the carbonate system;

Algorithm 4.1 Example of the model template file `General/GlobalDefsBFM.model` with the definition of pelagic state variables.

```
3d-state
  #-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
  #     State Variable(s) for Nutrients
  #-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
   N1p  :       Phosphate : mmol P/m3
   N3n  :       Nitrate  : mmol N/m3
   N4n  :       Ammonium : mmol N/m3
   O4n  :       NitrogenSink : mmol N/m3
   N5s  :       Silicate     : mmol Si/m3
   N6r  :       Reduction Equivalents : mmol S--/m3
 #-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
  #     State Variable(s) for Bacteria
  #-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
   B1[cnp]      : Pelagic Bacteria : mg C/m3 : mmol N/m3 : mmol P/m3
  #-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
  #     State Variable(s) for Phytoplankton Species
  #-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
     group PhytoPlankton[cnpls] :  mg C/m3 : mmol N/m3 : mmol P/m3 : mg Chl/m3 : mmmol Si/m3
        P1      : Diatoms
        P2[-s]  : Flagellates
        P3[-s]  : PicoPhytoPlankton
        P4[-s]  : Dinoflagellates
     end

  #-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
  #     State Variable(s) for Mesozooplankton
  #-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
     group MesoZooPlankton[cnp] : mg C/m3 : mmol N/m3 : mmol P/m3
                Z3[-np]         : Carnivorous mesozooplankton
                Z4[-np]         : Omnivorous mesozooplankton
     end
  #-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
  #     State Variable(s) for Microzooplankton
  #-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
     group MicroZooPlankton[cnp] : mg C/m3 : mmol N/m3 : mmol P/m3
                Z5 : Microzooplankton
                Z6 : Heterotrophic nanoflagellates (HNAN)
     end
end
```

INCLUDE_BEN Defines the benthic system variables;

INCLUDE_BENCO2 Defines the benthic variables of the carbonate system;

INCLUDE_BENPROFILES Activates the diagnostics for computing the vertical concentration of pore-water sediment components;

INCLUDE_SEAICE Defines the variables of the sea-ice biogeochemistry model (still experimental).

4.2.1 Example 1. Adding a subgroup

This is the most simple example, because it involves a small number of changes. Let us assume we want to add another mesozooplankton component to the zooplankton group, such as macrozooplankton. It is also assumed that this group will have exactly the same dynamics as the other two subgroups, but different parameter values. The first requirement is to edit the appropriate line in the template file `General/GlobalDefsBFM.model`, by adding the new variable name and long name:

Algorithm 4.2 Example of the model template file `General/GlobalDefsBFM.model` with the definition of pelagic state variables.

```
2d-state
  #-------------------------------------------------------------------
  # Benthic State Vars
  #-------------------------------------------------------------------
    group BenOrganisms[cnp] :mg C/m2:mmol N/m2:mmol P/m2
                Y1      : Epibenthos
                Y2      : Deposit feeders
                Y3      : Suspension feeders
                Y4      : Meiobenthos
                Y5      : Benthic predators
    end
    Q6[cnps]    :Particulate organic carbon  :mg C/m2:mmol N/m2:mmol P/m2:mmolSi/m2
    group BenDetritus[cnp] :mg C/m2:mmol N/m2:mmol P/m2
                Q1      : Labile organic carbon
                Q11     : Labile organic carbon
    end
    group BenBacteria[cnp]:mg C/m2:mmol N/m2:mmol P/m2
                H1                      : Aerobic benthic bacteria
                H2                      : Anaerobic benthic bacteria
    end
  #-------------------------------------------------------------------
  #     Benthic nutrient dynamics
  #-------------------------------------------------------------------
    group BenthicPhosphate[p] :mmol P/m2
                K1              : Phosphate in oxic layer
                K11             : Phosphate in denitrification layer
                K21             : Phosphate in anoxic layer
    end
    group BenthicAmmonium[n] :mmol N/m2
                K4              : Ammonium in oxic layer
                K14             : Ammonium in denitrification layer
                K24             : Ammonium in anoxic layer
    end
    K3[n]       : Nitrate in sediments :  mmol N/m2
    K5[s]       : Silicate in sediments : mmol Si/m2
    K6[r]       : Reduction equivalents in sediments : mmolS--/m2
    G2[o]       : Benthic O2 : mmol O2/m3
    G4[n]       : N2 sink for benthic system. :mmol N/m2
    D1[m]       : Oxygen penetration depth      :m
    D2[m]       : Denitrification depth         :m
    D6[m]       : Penetration Depth organic C   :m
    D7[m]       : Penetration Depth organic N   :m
    D8[m]       : Penetration Depth organic P   :m
    D9[m]       : Penetration Depth organic Si  :m
end
```

```
#-------------------------------------------------------------------------
#     State Variable(s) for Mesozooplankton
#-------------------------------------------------------------------------
     group MesoZooPlankton[cnp] : mg C/m3 : mmol N/m3 : mmol P/m3
                Z2         : Macrozooplankton
                Z3         : Carnivorous mesozooplankton
                Z4         : Omnivorous mesozooplankton
     end
```

The second step is the addition of a call for this new group in `PelagicSystemDynamics`, exactly as already done for Z3 and Z4:

```
if ( CalcMesoZooPlankton(iiZ2)) then
    call MesoZooDynamics( iiZ2, ppZ2c, ppZ2n, ppZ2p)
end if
if ( CalcMesoZooPlankton(iiZ3)) then
    call MesoZooDynamics( iiZ3, ppZ3c, ppZ3n, ppZ3p)
end if
if ( CalcMesoZooPlankton(iiZ4)) then
    call MesoZooDynamics( iiZ4, ppZ4c, ppZ4n, ppZ4p)
end if
```

The third step is temporarely needed, but will probably be modified and made automatic in one of the next release. It involves the creation of variables carrying the initial values to be read in the `bfm.nml` namelist (`bfm_init_nml`) from file `share/init_var_bfm.F90`. One for each component need to be created (in this case `Z2c0`, `Z2n0` and `Z2p0`), in all the places where the variables for the other members of the group are defined.

Now the compilation can be done and the script will take care of the automatic generation of the new memory layout with the required named constants and pointers. Check the updated file `General/ModuleMem.F90` and search for the new variable Z2 to see the added pieces of code.

The final step before running the new model is the addition of a new parameter column in `MesoZoo.nml`, with the required new parameter values.

4.2.2 Example 2. Adding a biochemical component and a group

The BFM application in the global ocean is called PELAGOS (PELAgic biogeochemistry for Global Ocean Simulations) and the climatological simulations were described in Vichi et al. (2007a). A major requirement to simulate the global ocean ecosystem is the introduction of iron control on phytoplankton growth, particularly to capture the high-nutrient low-chlorophyll regions. The example below shows how the iron component of phytoplankton and detritus can be added to the model structure template in order to prepare the memory settings. Additional coding from the user is then required to parameterize the fluxes between these components and the regulations of plankton physiology.

For a description of the iron parameterizations used in PELAGOS, see Vichi et al. (2007a). It is assumed that iron is not contained in zooplankton and bacteria, therefore, the dissolved organic iron release from zooplankton is directly done in particulate form and a first-order remineralization rate is implemented to parameterize bacterial syderophore production.

To add an iron constituent to a group, it is sufficient to specify a unique single-character constant between the square brackets in the `GlobalDefsBFM.model` file as shown below:

```
#-------------------------------------------------------------------------
#     State Variable(s) for Phytoplankton Species
#-------------------------------------------------------------------------
  group PhytoPlankton[cnplsf] :  mg C/m3 : mmol N/m3 : mmol P/m3 :
                                 mg Chl/m3 : mmmol Si/m3 : umol Fe/m3
        P1      : Diatoms
        P2[-s]  : Flagellates
        P3[-s]  : PicoPhytoPlankton
```

```
      P4[-s]  : Dinoflagellates
end
```

It is also necessary to provide units separated by colons, in this case μmol Fe m^{-3}. The `[-s]` syntax in some phytoplankton groups indicates that this constituent, although being a component of the group vector, is not computed for that specific sub-group.

The preprocessing script will take care during compilation to generate an adequate entry in the main memory module and the definitions of the named constants (`ppP1f`,...) and pointers (`P1f`, ...) to access the new variables from the model code.

New groups can also be defined in order to use the facilities shown in Sec. 3.4.3. All the detritus variables can be grouped together with the following syntax:

```
#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
#     State Variable(s) for Detritus (Biogenic Organic Material)
#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
  group PelDetritus[cnpsf] : mg C/m3 : mmol N/m3: mmiol P/m3: mmol Si/m3:
                             umol Fe/m3
      R1[-sf]    : Labile Organic Carbon (LOC)
      R2[-npsf]  : Carbohydrates
      R6         : Particulate Organic Carbon (POC)
      R7[-npsf]  : Refractory DOC
  end
```

4.2.3 Example 3. Removing components from zooplankton

For certain purposes it is not needed to carry on the computation of variable nutrient:carbon quota in selected Living Functional Groups (for instance, in the case of the new macrozooplankton gorup added in example 1). This can be due to save computational resources in case of low-resolution long-term global ocean simulations or with high-resolution coastal models. A possible example is the assumption of fixed quota in zooplankton which allows to reduce the number of standard components of up to 8 state variables.

The template for this example is available in `BFM/General/Configurations/ GlobalDefsBFM.model` and should replace the standard `GlobalDefsBFM.model`. The zooplankton group definitions are equal to the one shown in the code of Box 4.1 for MesoZooplankton. The additional requirement is the change of value in parameter check_fixed_quota from 0 to 1 in the Param.nml namelist. Test experiments have shown that the simulation times can be reduced up to 30%. The physiological assumption is that zooplankton release in inorganic form the nutrients in excess of the given fixed quota.

4.3 Parallel computing

The BFM has no internal parallelization structure but it is built to be easily parallelized following the technique implemented in the transport model. Since the grid domain is stripped of all the land points (BFM memory layout is a 1-D array), the BFM has no need to know the global domain structure and is thus defined for any subdomain assigned to a process/processor.

The only working implementation tested at the moment is with NEMO and the MPI protocol. The adopted strategy was to use the local domain indices as the numbers defining the 3-D structure from which the ocean points are extracted. Thus, the number of BFM ocean points is equal to the number of ocean points in each subdomain.

A parallel simulation implies a special way of handling the output, restart and log files which are generated by each process. The addition of the macro `BFM_PARALLEL` during the compilation specifies a different name for these files by appending a 4 digit integer at the end of the name, according to the rank of the process that generates them (e.g `bfm_0000.nc` for the output of

process rank 0 or `bfm_restart_0001.nc` for the restart file of rank 1 processor). Note that the rank of the process has to be provided to the BFM by the transport model through the variable `parallel_rank` in the module `api_bfm` (in NEMO, for instance, this assignement is done in the routine `ini_trc_bfm.F90`). By default each process generates its own log file, but it is possible to have all the information written by the rank 0 process only by setting to true the logical flag `parallel_log` in the bfm.nml namelist file.

5 Diagnostics

5.1 Output

The variables that can be stored in the output file are the ones defined in `GlobalDefsBFM.model`. A list of the exact names to be used is reported in comment lines in the auto-generated file `ModuleMem.F90` as shown in Box 5.1.

All the state variables can be stored (provided that they have a long name and units), both as instantaneous values and as means over the chosen output period. This behavior is controlled via the namelist `bfm.nml` when running the STANDALONE model. Other couplings may have different names. Additionally, a set of other variables can be saved, such as the ones obtained from the physical model and other diagnostics such as total chlorophyll and the nutrient quota inside the LFGs. The system cannot automatically store 0D variables (with only time dimension defined) .

5.2 Restart

The length of each BFM simulation is controlled by the physical transport model except for the STANDALONE model which has its own time marching. It is common practice in modelling to split the simulation in several chunks and to continue the simulation from the end values of each state variable at the end of a previous run. The BFM stores this information in a NetCDF file which is always generated at the end of each simulation. There is no automatic control on the date (neither it is stored as an attribute inside the NetCDF file), so it is totally up to the user to check that the initial restart file is the proper one.

5.3 Aggregated diagnostics and rates

All the ancillary 2D and 3D variables and diagnostics are grouped in two memory arrays, `D3DIAGNOS` and `D2DIAGNOS`, accessed by means of pointers. The dimensions of these arrays change according to the number of model elements in the `GlobalDefsBFM.model` file, and any diagnostic rate can be easily defined thanks to the automatic generation of the memory layout. Some fluxes are already defined by default, such as the gross/net primary production rates, total respiration rates and (de)nitrification rates. Any exchange of mass between every state variable can be tracked by means of a special syntax that access and combines the elements of the `D3(D2)SOURCE` and `D3(D2)SINK` arrays where all the rates are stored. Some examples are already present at the end of the `GlobalDefsBFM.model` file and are activated with the compilation macros `INCLUDE_DIAG3D` and `INCLUDE_DIAG2D`:

```
fP1Z4c=P1c->Z4c: diatom grazing by omnivorous zooplankton: mg C/m3/d
ruPTc=P.c<-*: gross primary production: mg C/m3/d
ruZTc=(Z.c<-P.c+B1c+Z.c)-(Z.c->R1c+R6c): gross secondary production: mg C/
```

The first line defines a diagnostic storing the carbon flux between diatoms and mesozooplankton. This diagnostic also defines the long_name and units attributes of the NetCDF and it can thus be

Algorithm 5.1 Examples of auto-generated comment lines in ModuleMem.F90 reporting the variables that can be stored. The list depends on the definitions in `GlobalDefsBFM.model`

```
!-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
!! GLOBAL definition of Pelagic (D3) variables which can be stored
!-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
!    3d name                                                  description           unit
! ---------- ------------------------------------------------------------ ---------------
!        ETW                                                  temperature              C
!        ESW                                                     Salinity             --
!        EIR                                                   Irradiance       uE/m2/s
!        ESS                                           Suspended sediment           g/m3
!       ERHO                                                      density           g/m3

!      cxoO2                       Temperature-dependent oxygen saturation      mmol O2/m3
!       XO2o                                       Net O2 conc. = O2 - H2S      mmol O2/m3
!     eO2mO2                                    Relative Oxygen saturation               %
!       Chla                                                   Chlorophyl      mg Chl/m3
! qsPc(iiP1)                              Quotum si/c in P4(PhytoPlankton)    mmol Si/mg C
! qlPc(iiP1)                             Quotum chl/c in P1(PhytoPlankton)    mg Chl /mg C
! qlPc(iiP2)                             Quotum chl/c in P2(PhytoPlankton)    mg Chl /mg C
! qlPc(iiP3)                             Quotum chl/c in P3(PhytoPlankton)    mg Chl /mg C
! qlPc(iiP4)                             Quotum chl/c in P4(PhytoPlankton)    mg Chl /mg C
! qpZc(iiZ3)                                Quotum p/c Z3(MesoZooPlankton)     mmol P/mg C
.....
!-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
!!  GLOBAL definition of Benthic (D2) variables which can be outputted in netcdf
!-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
!    2d name                                                  description           unit
! ---------- ------------------------------------------------------------ ---------------
!      rrBTo                                 Total Benthic oxic respiratio    mmol O2/m2/d
!      rrATo                              Total Benthic anoxic respiration    mmol O2/m2/d
!      reBTn                                 Total Benthic oxic respiratio     mmol N/m2/d
!      reBTp                                 Total Benthic oxic respiratio     mmol P/m2/d
!      reATn                              Total Benthic anoxic respiration     mmol N/m2/d
!      reATp                              Total Benthic anoxic respiration     mmol P/m2/d
!     turenh                       Enhancement factor due to bioirrigation               -
!     irrenh                        Enhancement factor due to bioturbation               -
!   shiftD1m                                        Rate of change of D1m              m/d
!   shiftD2m                                        Rate of change of D2m              m/d
!     jG2K3o                          Oxygen consumption by nitrification.    mmol O2/m2/d
!     jG2K7o                        ReOxidation of Red.Equiv. in  oxic layer   mmol S--/m2/d
!        M1p                                     phosphate in oxic layer        mmol P/m3
!       M11p                             phosphate in denitrification layer       mmol P/m3
!       M21p                                   phosphate in anoxic layer        mmol P/m3

...............
```

included in the output just by adding it to the `bfm_save_nml` section in `bfm.nml`. The second line defines the combined rate gross primary production, which is the sum of all the carbon fluxes to all the phytoplankton groups (notice the syntax without the subgroup number). The last example is the gross secondary production, which is defined as the difference between the carbon ingestion by all zooplankton groups and the carbon loss to dissolved and particulate detritus.

Boundary fluxes are defined by default for each pelagic state variable. The variables

```
PELSURFACE(NO_D3_BOX_STATES,NO_BOXES_XY)
PELBOTTOM(NO_D3_BOX_STATES,NO_BOXES_XY)
```

are themselves pointers to the main diagnostic memory `D2DIAGNOS`. They can be accessed by means of specific pointers allocated in the generated subroutine `BFM/General/AllocateMem.F90` (notice that the integer constants change at every code generation and are only reported here as an example of the code):

```
PELSURFACE => D2DIAGNOS(17+1:17+50,:); PELSURFACE=ZERO
jsurO2o => D2DIAGNOS(17+ppO2o,:); jsurO2o=ZERO
jsurN1p => D2DIAGNOS(17+ppN1p,:); jsurN1p=ZERO
jsurN3n => D2DIAGNOS(17+ppN3n,:); jsurN3n=ZERO
jsurN4n => D2DIAGNOS(17+ppN4n,:); jsurN4n=ZERO
```

The boundary rates (always in units per day) are accessed in the code by means of the PELSURFACE(ppO2o,:) or jsurO2o(:) names.

5.4 Mass conservation

The total mass in the system (mass units m^{-2}) can be checked by means of built-in diagnostic variables. There is a variable for each biogeochemical component (C, N, P, Si, etc.) both for the pelagic and the benthic systems. To activate the diagnostic it is necessary to change the setup to benthic-pelagic (`bio_setup=3`), turn on/off the benthic system if needed (`CalcBenthicFlag` in `Param.nml`) and then add the following lines to the output namelist in the `var_save` section:

```
var_save =
                  'totpeln',
                  'totpelp',
                  'totpels',
                  'totbenn',
                  'totbenp',
                  'totbens'
```

The actual computation is done in `General/CheckMassConservation.F90`. Note that for technical reasons the total mass variables are multidimensional but the value is stored in the first element of the array.

When checking a coupling with an OGCM, it is also important to turn off any additional boundary flux (e.g. dilution, atmospheric deposition, river input, etc.) because they are not taken into account by the standard BFM.

6 The coupling with NEMO

This chapter describes the coupling of the BFM with NEMO, both for the one-dimensional setup and for the full three-dimensional implementation.

This is the current new release of the PELAGOS model described in Vichi et al. (2007b,a). The 1-D configuration is activated through the pre-processor macro `key_cfg_1d` during compilation, and the coupling with BFM is done within the passive tracer implementation (macro key_passivetrc) with the additional key `key_trc_bfm`.

6.1 The original NEMO structure

NEMO is quite a complex model from the structural perspective because of the many macros which activate different modules. The original structure is shown in Fig. 6.1, where we see both the call to the 1-D and 3-D timestepping.

The tracer model shares the main memory with OPA (domain characteristics, coordinate types, time and output manager parameters, etc.) through a module collector `trc_oce`, which acts as a transparent interface layer. This reduces the number of modules to be use-associated inside the tracer subroutines because the use of trc_oce allow to access all the needed variables form the other modules (Fig. 6.2). On the other hand, this implies that the tracers subroutines that uses trc_oce have all the same shared memory. The required variable names are not renamed but only re-scoped through pointer association to simplify code writing.

The number of passive tracers (i.e. biogeochemical pelagic state variables) is defined in module `par_trc_trp(.F90)`, where it is possible to set the dimensions of the tracers memory arrays (tra, trb, trn) declared in module `trc(.F90)`. There is no dynamical allocation of the tracer memory, which poses some problems for the coupling with the BFM.

6.2 Coupling strategy

In the first PELAGOS implementation with OPA 8.2 the memory array was modified and reduced to one single temporary 3-D field, which was sequentially filled in with each BFM state variable to compute the advection and diffusion trends.

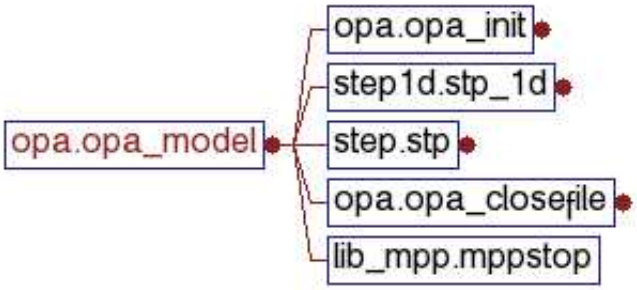

Figure 6.1: The NEMO main invocation tree

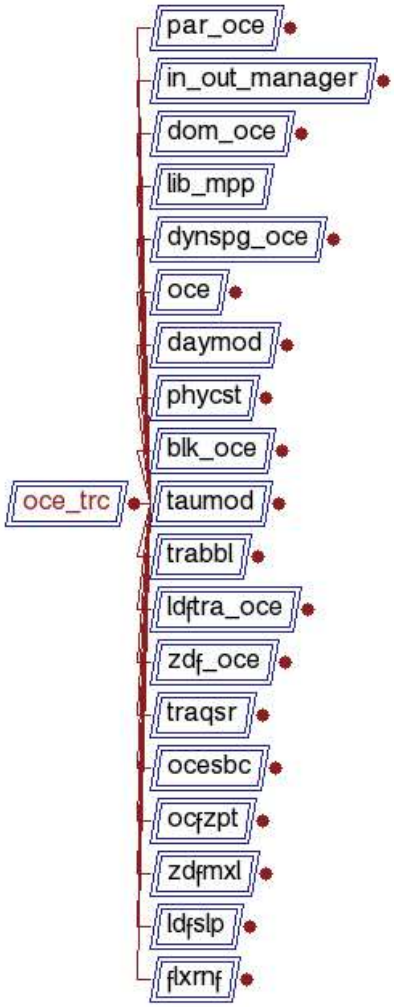

Figure 6.2: List of the modules which are use-associated within trc_oce.

The basic idea is to maintain the current NEMO SMS structure as much as possible. However, it is not possible to share the memory between the tracer module and the BFM due to the different shapes of array (pointer association only works along one dimension). Therefore, the likely solution is still to keep the original memory layout but reduce it to one single tracer and loop over the number of state variables.

There are 2 different macros that needs to be activated, one for OPA (`key_trc_bfm`) and one for BFM (`BFM_NEMO`).

6.3 Changes to NEMO

6.3.1 version 1_12 (May 2006)

The work has been initially carried out in a devlopment branch of the original nemo source code (version 1_12), extracted from the CVS server and ported to the subversion repository (`http://www.bo.ingv.it/svn/nemo/trunk`).

Changes were done in the TOP_SRC directory, and the modified files are the following:

`NEMO/TOP_SRC/initrc.F90`
`NEMO/TOP_SRC/oce_trc.F90`
`NEMO/TOP_SRC/par_trc_trp.F90`
`NEMO/TOP_SRC/trcctl.F90`
`NEMO/TOP_SRC/trclsm.F90`
`NEMO/TOP_SRC/trcsms.F90`
`NEMO/TOP_SRC/TRP/trcstp.F90`
`NEMO/TOP_SRC/TRP/trctrp_lec.F90`

6.3.2 version 2 (November 2006)

The changes only involve the OPA dynamical core and the coupling with the SMS is mostly untouched.

6.3.3 Initialization procedures

The standard NEMO SMS initialization is carried out in opa_init with a call to ini_trc. BFM is initialized in subroutine trc_ini_bfm, which belongs to the BFM tree.

6.3.4 The tracer timestepping

Tracers are computed inside the main time-loop with a call to subroutine `trc_stp` (Fig. 6.4, called by `stp_1D` and `stp` in the 1D and 3D configurations, respectively). The standard Source-minus-Sink routine has been modified to calll

6.4 New BFM subroutines

The additional files needed to compute the BFM source terms and to transport the biogeochemical variables are found in the BFM directory `$BFMDIR/src/nemo`.

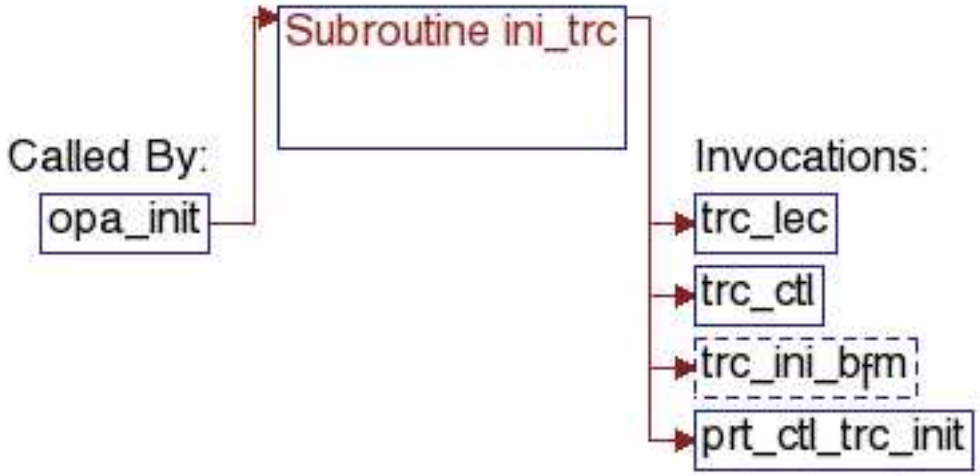

Figure 6.3: Invocation of the initialization routine for the BFM tracers. BFM routines are indicated with dashed boxes.

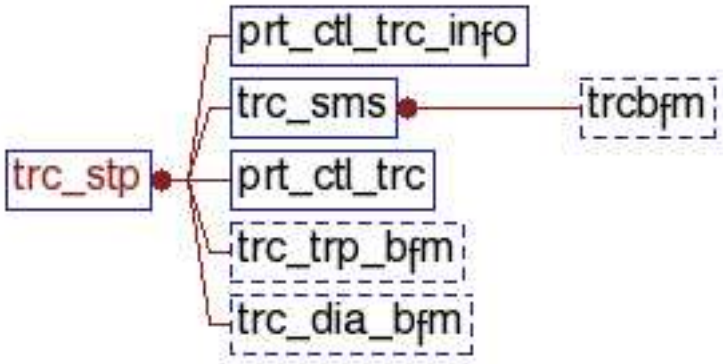

Figure 6.4: Invocation of the timestepping routine for the BFM tracers. BFM routines found in `$BFMDIR/src/nemo` are indicated with dashed boxes.

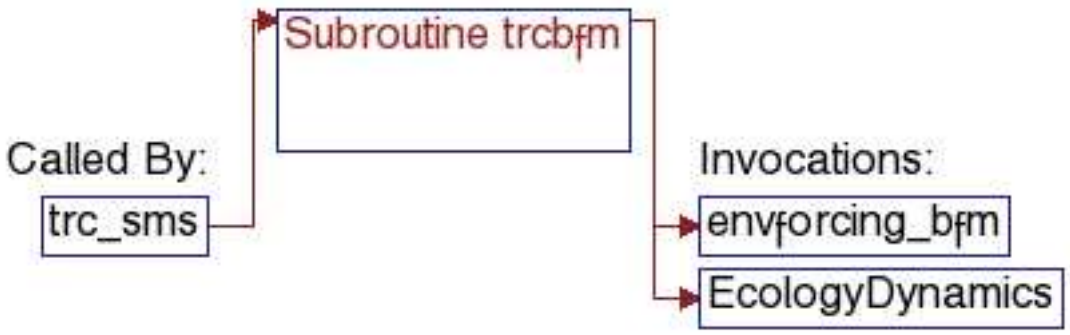

Figure 6.5: Invocation of the BFM dynamical core. All these routines belong to the `$BFMDIR/src/nemo` directory.

6.4.1 BFM reaction terms

The computation of the reaction terms is done in `trcbfm.F90`, where the physical variables are first transferred from OPA to the BFM memory in `envforcing_bfm.F90` and then the call to the standard `EcologyDynamics` starts the BFM dynamical core (Fig. 6.5). It is also possible to solve the BFM equations as a set of ODE with a simple Euler-forward solver when the `CalcTransportFlag` is set to `FALSE` in the main parameter namelist `Param.nml`. This switch excludes the computation of the transport terms described in the next section and let the BFM behave as a set of unconnected boxes while the physical model is normally computed. The benthic components, if activated, are always solved with a simple Euler-forward solver in the current version (see the GOTM chapter for a description of other more sophisticated numerical solvers that can be implemented).

6.4.2 BFM transport terms

The core of the physical transport is computed in `trc_trp_bfm` (Fig. 6.4). Here the BFM state variables are mapped onto the 3D temporary array for evaluating the transport terms (cfr. Sec.). The routine `trc_set_bfm` computes the additional boundary conditions (in `trc_sbc` only the dilution factor is computed) and adds the advection term due to sinking using a conservative first-order scheme. Internal damping to observed fields is not implemented for the BFM.

The different advection and diffusion schemes due to the water dynamics are then applied according to the choices of the `passivetrc.namelist` parameters. At the end, the resulting forward-in-time 3D field is re-mapped onto the 1D memory of the BFM.

6.5 Diagnostics

The BFM uses its own storage routines to save the diagnostics in order to preserve the major features (see Chap. 5). The routine `trc_dia_bfm` (Fig. 6.4) is a wrapper for the `save_bfm` standard routine. The name of the NetCDF output file and the frequency of storage is defined in the `bfm.nml` namelist.

6.6 Compilation

The compilation environment of NEMO is scattered in various scripts and subdirectories. It allows the compilation for a large set of architectures, which implies the need to generate the Makefiles

all the time. The BFM is not directly included in the NEMO tree, but is kept separate and identified through the definition of the environmental variable BFMDIR that points to the root directory of the local BFM version. This variable needs to be mandatorily defined before the construction of the Makefile with the script `UTIL/fait_AA_make.`

In order to integrate the BFM whithin this compilation environment and perform the control over the number of compilation macros, the scripts contained in the `UTIL` directory have been modified. A new configuration ORCA2_LIM_BFM_1D has been added in `fait_AA_make`. The new pieces of code in `fait_AA_make` adds the lines in the final Makefile of the `WORK` directory that are required to activate the BFM compilation when the macro `key_trc_bfm` is used.

The following summarizes the steps required to compile NEMO with the BFM in a 1D configuration:

1. Download the BFM in a separate directory and set the environmental variable `BFMDIR` to that directory
2. Prepare the configuration:
   ```
   cd modeles/UTIL
   ./fait_config ORCA2_LIM_BFM_1D
   ```
3. edit the global definitions in `AA_make.gdef` according to your own platform
4. create the `AA_make` files
   ```
   cd modipsl/modeles/NEMO
   ../UTIL/fait_AA_make
   ```
5. insert the makefiles for your architecture (e.g. for linux with compiler ifort)
   ../../util/ins_make -v -d -p I4R8 -t lxiv8
6. ```
   cd WORK
   make
   ```

The BFM compilation will start at the end of the compilation of OPA and the used Makefile is the one in `$BFMDIR/src/nemo`. The major expected problems are related to the link with the netcdf library which needs to be compiled with the same options defined for OPA9. Notice that in the default NEMO compilation environment, the final compilation options are set in the `BB_make` file of the specific configuration and not in the `AA_make.gdef` as one may expect.

6.7 Test case simulation
   ```

Bibliography

Baretta, J., Ebenhöh, W., Ruardij, P., 1995. The European Regional Seas Ecosystem Model, a complex marine ecosystem model 33 (3-4), 233–246.

Baretta-Bekker, J., Baretta, J., Ebenhoeh, W., 1997. Microbial dynamics in the marine ecosystem model ERSEM II with decoupled carbon assimilation and nutrient uptake 38 (3/4), 195–212.

Blackford, J., Radford, P., 1995. A structure and methodology for marine ecosystem modelling 33 (3-4), 247–260.

Ebenhöh, W., Baretta-Bekker, J., Baretta, J., 1997. The primary production module in the marine ecosystem model ERSEM II with emphasis on the light forcing 38, 173–193.

Ruardij, P., Baretta, J., Baretta-Bekker, J., 1995. SESAME, a Software Environment for Simulation and Analysis of Marine Ecosystems 33 (3-4), 261–270.

Vichi, M., Masina, S., Navarra, A., 2007a. A generalized model of pelagic biogeochemistry for the global ocean ecosystem. Part II: numerical simulations 64, 110–134.

Vichi, M., Pinardi, N., Masina, S., 2007b. A generalized model of pelagic biogeochemistry for the global ocean ecosystem. Part I: theory 64, 89–109.