# Coupling BFM with ocean models:

## the 1D Princeton Ocean Model

***M. Zavatarelli, N. Pinardi, G. Mussap, T. Lovato, C. Amadio, L. Mentaschi, M Butenschön, M. Vichi***

Release 1.1, February 2023
—— BFM Report series N. 3 ——

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

BFM
BIOGEOCHEMICAL FLUX MODEL

The BFM system team is gratefully indebted with Prof. George L. Mellor for allowing us to distribute the FORTRAN95 computer code of the one-dimensional Princeton Ocean Model that evolved from the original FORTRAN77 version of the model.

This document should be cited as:
Zavatarelli M., Pinardi N., Mussap G., Lovato T., Amadio C., Mentaschi L., Butenschön M., Vichi M. (2023). Coupling BFM with Ocean models: the 1D Princeton Ocean Model. BFM Report series N. 3, Release 1.1, February 2023, Bologna, Italy, http://bfm-community.eu, pp. 58

# Contents



[1] Most of this section is taken from Mellor (2004)
[2] Most of this section is taken from Mellor (2004)

# 1 Introduction

This report describes the online coupling between the one-dimensional version of the Princeton Ocean Model (hereafter POM1D), developed by Blumberg and Mellor (1987), and the Biogeochemical Flux Model (hereafter BFM), in term of the main equations and their translation into a computer code, and its code is distributed with BFM (see details in http://bfm-community.eu).

The report is organised as follows:

Section 2 describes the governing equations and the boundary conditions used in the POM1D to simulate the ocean physical dynamics. A full description of biogeochemical equations is given in the BFM core manual Vichi et al. (2022).

Section 3 is devoted to the description of the general characteristics of the BFM-POM1D coupling. The coupled equations and the relative boundary conditions are described along with the equation transformation coherent with the vertical coordinate system adopted. The section closes with an overview of the discretized equations and of the numerical technique and scheme used to solve them.

Section 4 provides detailed description and information of the BFM-POM1D computer code. The `main program`, the `modules` and the `subroutines` composing the system are detailed with reference to the governing equations and information about practical use and specific implementation modifications are provided in order to facilitate the use.

Section 5 gives information about the installation of the modeling system, while section 6 shows results from simulation of the BFM-POM1D system implemented in the Gulf of Trieste. Initial conditions and forcing functions for such experiments are provided by the BFM site, so that an interested user can check its installation by replicating such results.

# 2 POM1D physical model

## 2.1 General

POM1D is the one-dimensional version of the three dimensional Princeton Ocean Model, POM (Blumberg and Mellor, 1987), a primitive equation ocean circulation model formulated in sigma coordinates. Its prognostic state variables are the velocity, temperature, salinity and turbulent kinetic energy fields. Diffusivity is computed by mean of the second order turbulence closure scheme proposed by Mellor and Yamada (1982). A full description of the original three dimensional model equations and the numerical scheme corresponding to the 1D implementation applied here, can be found in Blumberg and Mellor (1987) and in Mellor (2004).

## 2.2 The Governing equations

POM1D originates directly from a modification of the original three-dimensional model. The model adopts the hydrostatic and the Boussinesq approximation. The one-dimensional primitive equations and the equations for the physical tracers, written only for the $z$ (vertical space) and $t$ (time) independent variables are:

$$\frac{\partial (u,v)}{\partial t} \mp f(v,u) = \frac{\partial}{\partial z}\left[K_M \frac{\partial (u,v)}{\partial z}\right] + F_u \tag{2.2.1}$$

$$\frac{\partial p}{\partial z} = -\rho g$$

$$\frac{\partial w}{\partial z} = 0$$

$$\frac{\partial T}{\partial t} = \frac{\partial}{\partial z}\left(K_H \frac{\partial T}{\partial z}\right) + \Phi_T + F_T - \frac{\partial R}{\partial z} \tag{2.2.2}$$

$$\frac{\partial S}{\partial t} = \frac{\partial}{\partial z}\left(K_H \frac{\partial S}{\partial z}\right) + \Phi_S + F_S \tag{2.2.3}$$

$$\frac{\partial q_2}{\partial t} = \frac{\partial}{\partial z}\left(K_H \frac{\partial q_2}{\partial z}\right) + \frac{2g}{\rho_0} K_H \frac{\partial \widetilde{\rho}}{\partial z} - \frac{2q_2^3}{B_1 l} + F_q \tag{2.2.4}$$

$$\frac{\partial q_2 l}{\partial t} = \frac{\partial}{\partial z}\left(K_Q \frac{\partial q_2 l}{\partial z}\right) + \frac{l E_1 g}{\rho_0} K_H \frac{\partial \widetilde{\rho}}{\partial z} - \frac{q_2^3}{B_1}\left[\!\!\left[1 - \left\{E_2\left[\frac{lH}{\kappa z (H+z)}\right]^2\right\}\right]\!\!\right] + F_l \tag{2.2.5}$$

Where $u$, $v$ $(m/s)$ are the mean velocity components; $T$(ºC) and $S(psu)$ are the mean potential temperature and salinity vertical profiles; $H$ $(m)$ is the bottom depth; $f$ is the Coriolis parameter; $p$

$\left(N/m^2\right)$ is the pressure; $\rho$, $\rho_0$ $\left(kg/m^3\right)$ are the instantaneous and reference seawater density respectively. The solar radiation, $R$ ($^{\circ}Cm/s$), reaching the sea surface and penetrating the water column, is vertically attenuated following a double exponential decay relationship, parameterized according to Paulson and Simpson (1977):

$$R = (\rho_0 C_p)^{-1} Q_s \left[ R_p e^{\frac{z}{\zeta_1}} + (1 - R_p) e^{\frac{z}{\zeta_2}} \right] \tag{2.2.6}$$

and is added to the heat equation 2.2.2 as $\partial R/\partial z$ . In eq. 2.2.6 $Q_s$ is the surface solar radiation flux $(W/m^2)$, $C_p$ is the seawater specific heat, while the Radiance partition coefficient ($0 \leq R_p \leq 1$) and the attenuation lengths ($\zeta_1$and $\zeta_2$ in m) coefficients can be selected from the Jerlov (1976) water (optical) types.

Twice the kinetic energy ($q_2$), twice the kinetic energy times the turbulence length scale $l$ ($q_2 l$) and the turbulence vertical diffusivities $K_M$, $K_H$ and $K_Q$ $\left(m^2/s\right)$ are provided by the Mellor and Yamada (1982) 2.5 turbulence closure scheme. In eqs. 2.2.4 and 2.2.5 $B_1$ $E_1$ and $E_2$ are turbulence closure parameters (see sec. 4.3.14); $\kappa = 0.4$ is the von Karman constant and $\partial\widetilde{\rho}/\partial z = (\partial\rho/\partial z) - c_s^{-2}(\partial p/\partial z)$,with $c_s = c_s(T,S)$ being the speed of sound. $\Phi_{(T,S)}$ are the terms that parameterize lateral advection of temperature and salinity, and $F_{(u,v,q,l,T,S)}$, parameterize unresolved diffusive processes. They are written likewise the turbulent diffusion terms:

$$F_{(u,v,q)} = \frac{\partial}{\partial z}\left[\chi_M \frac{\partial (u,v,q)}{\partial z}\right] \tag{2.2.7}$$

$$F_l = \frac{\partial}{\partial z}\left(\chi_L \frac{\partial q_2 l}{\partial z}\right) \tag{2.2.8}$$

$$F_{(T,S)} = \frac{\partial}{\partial z}\left[\chi_{(T,S)} \frac{\partial (T,S)}{\partial z}\right] \tag{2.2.9}$$

where $\chi_{(M,l,T,S)}$ $\left(m^2/s\right)$ are background diffusion coefficients.

## 2.3 The surface and bottom boundary conditions

The surface boundary conditions (applied at the depth of surface-most gridpoint, $z_s$) for the momentum equations is the following:

$$\left[(K_M + \chi_M)\frac{\partial (u,v)}{\partial z}\right]_{z=z_s} = \frac{\tau^W_{(x,y)}}{\rho_0} \tag{2.3.1}$$

while those for $q_2$ and $q_2 l$ are :

$$q_{2|z=z_s} = (B_1)^{2/3}\left[\left(\frac{\tau^W_{(x)}}{\rho_0}\right)^2 + \left(\frac{\tau^W_{(y)}}{\rho_0}\right)^2\right]$$

$$q_2 l\,|_{z=z_s} = 0$$

where $\tau^W_{(x,y)}$ $(N/m^2)$ are the (zonal and meridional) components of the wind stress. The wind stress field is an external forcing function to be provided to the model.

The momentum bottom boundary condition requires the computation of the bottom stress zonal and meridional components ( $\tau^B_{(x,y)}$). It is computed from the bottom current velocity (the velocity at the depth of the bottommost gridpoint, $z_b$):

$$\left[(K_M+\chi_M)\frac{\partial(u,v)}{\partial z}\right]_{z=z_b}=\frac{\tau^B_{(x,y)}}{\rho_0}=c_z\left\{\left[u^2+v^2\right]^{0.5}(u,v)\right\}_{z=z_b} \tag{2.3.2}$$

where $c_z$ is a quadratic bottom drag coefficient computed according to:

$$c_z=MAX\left\{\frac{\kappa^2}{\left[ln\left(\frac{H-z_b}{z_0}\right)\right]^2},c_{z_{min}}\right\} \tag{2.3.3}$$

where $z_0$ is the bottom roughness length $(m)$ and $c_{z_{min}}$ is the minimum achievable $c_z$ value.

The bottom boundary conditions for $q_2$ and $q_2l$ are :

$$q_2\mid_{z=zb}=(B_1)^{2/3}\left[\left(\frac{\tau^B_{(x)}}{\rho_0}\right)^2+\left(\frac{\tau^B_{(y)}}{\rho_0}\right)^2\right]^{0.5}$$

$$q_2l\mid_{z=zb}=0$$

The surface boundary condition for temperature is:

$$\left[(K_H+\chi_T)\frac{\partial T}{\partial z}\right]_{z=z_s}=(\rho_0C_p)^{-1}\left[\frac{\partial R}{\partial z}|_{z=z_s}-(Q_b+Q_e+Q_h)\right] \tag{2.3.4}$$

where $Q_b$ is the net long wave radiation flux, $Q_e$ is the latent heat flux and $Q_h$ is the sensible heat flux; (all of them in $W/m^2$).

The salinity surface boundary conditions is defined by a "virtual" salinity flux obtained by relaxing the surface salinity value to a prescribed time varying value:

$$\left[(K_H+\chi_S)\frac{\partial S}{\partial z}\right]_{z=z_s}=\alpha\left[S^*(0,t)-S(0,t)\right] \tag{2.3.5}$$

where $\alpha(\ m/s)$ is the relaxation velocity.

In alternative to the above surface boundary conditions for temperature and salinity, the POM1D code allows for prescribing a time varying surface temperature, $T^*(0,t)$ and salinity, $S^*(0,t)$ value:

$$[T(z_s,t),S(z_s,t)] = [T^*(0,t),S^*(0,t)] \tag{2.3.6}$$

At the ocean bottom an "adiabatic" boundary condition is applies:

$$\left[(K_H + \chi_{T,S})\frac{\partial(T,S))}{\partial z}\right]_{z=-H} = 0 \tag{2.3.7}$$

## 2.4 Diagnostic mode

POM1D structure has been modified (Bianchi et al., 2006, Mussap et al., 2016; 2017, Mussap and Zavatarelli, 2017) in order to allow for the performance of diagnostic simulations, achieved by prescribing climatological observed time dependent temperature and salinity vertical profiles. The vertical turbulent diffusion coefficients profiles are then computed by the Mellor and Yamada (1982) second order turbulent closure scheme on the basis of the prescribed (time varying) density vertical structure and of the wind dependent energy input. The coefficients are used to compute the time depending vertical profiles of the BFM state variables. The use of the "diagnostic" mode eliminates possible drifts in temperature and/or salinity due to the use of "non zero" surface heat and/or salinity surface fluxes or to the lack of a proper parametrisation of the lateral advective fluxes, which are, by construction, not contained in a one-dimensional model implementation. The use of the diagnostic mode with climatological data, provides a stable (non-drifting) annual cycle of the vertical density structure, which is particularly suitable when using the numerical model to investigate the coupled marine ecosystem dynamics. Clearly, the reliability of the simulations is crucially dependent on the quality of the assembled climatology.

# 3 BFM-POM1D physical biological coupling

## 3.1 The vertical coordinate system and the vertical grid arrangement.

The free surface $(\eta)$ three dimensional version of POM adopts a bottom following "sigma" vertical coordinate system: $\sigma = (z-\eta)/(H+\eta)$. In BFM-POM1D the vertical coordinate system reduces to a simple fractional coordinate system:

$$\sigma = \frac{z}{H} \tag{3.1.1}$$

with $-1 \leq \sigma \leq 0$ between ocean bottom and surface respectively.

The computational grid is vertically staggered. The sigma coordinate position of all the variables computed by the modeling system is schematized in Fig.3.1. $T$, $S$ $u$, $v$, $Q2$, $Q2L$, are all the variables computed (prognostically and/or diagnostically) by POM1D (see section 4.2.1). $C_p$ represents all the pelagic (water column) prognostic LFGs and CFFs variables, whose biogeochemical rate of change $(\partial C_p/\partial t)_{bgc}$ (see eq.3.2.1) is computed by BFM.

## 3.2 The coupled equation for the biogeochemical state variables.

### 3.2.1 The governing equations

The temporal total rate of change for a generic, pelagic (water column) non conservative BFM state LFG or CFF ($C_p$) can be written as follows:

$$\frac{\partial C_p}{\partial t} = \frac{\partial C_p}{\partial t}\,|_{phys} + \frac{\partial C_p}{\partial t}\,|_{bgc} \tag{3.2.1}$$

where the first term of the equation right hand side, indicates the rate of change dependent on the physical processes (handled by the BFM-POM1D coupling), while the second term indicates the rate of change dependent on the biogeochemical processes (handled by BFM). Therefore the physical rate of change is solved by the equation for a non conservative state variable to which, for selected LFGs and CFFs (such as phytoplankton and particulate organic detritus), a vertical advection term, accounting for sinking, must be added. The coupled equation then is a typical "advection, diffusion, reaction" equation (i.e. an equation dealing with the time evolution of chemical or biological species in a flowing medium such as water or air, Hundsdorfer and Verwer, 2003). and reads as:

$$\frac{\partial C_p}{\partial t} = \frac{\partial}{\partial z}\left[(K_H + \chi_b)\frac{\partial C_p}{\partial z}\right] + \frac{\partial (w_s C_p)}{\partial z} + \frac{\partial C_p}{\partial t}\,|_{bgc} \tag{3.2.2}$$

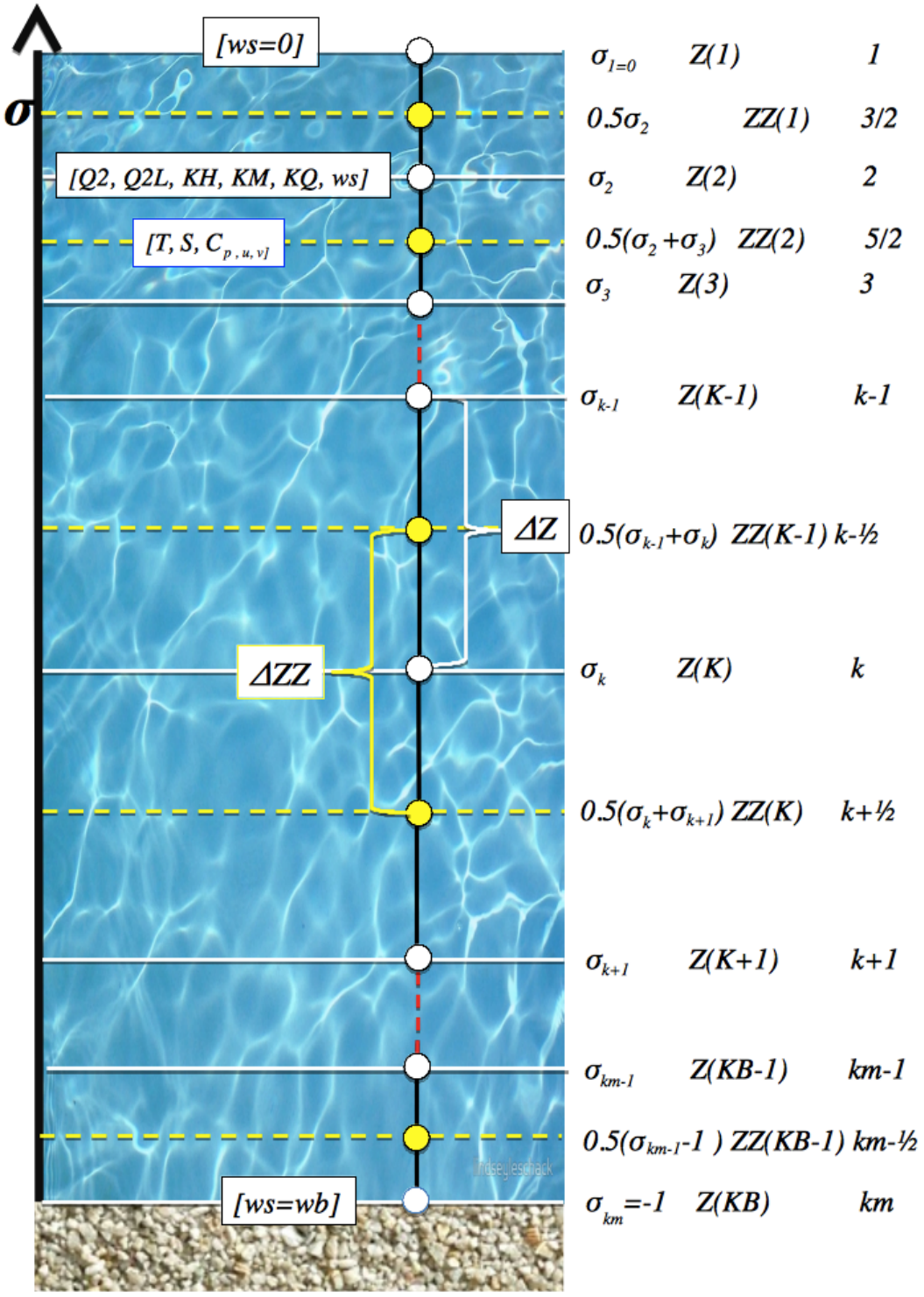


Figure 3.1: The vertically staggered grid of BFM-POM1D. The picture reports also the computation location of the System state variables. The upper case "K" indexing refers to the computer code indexing, while the lower case "k" indexing refers to the notation used for the equations in discrete form reported in the text. $k_m$ is the total number of model layers

where $\chi_b\left(m^2/s\right)$ is the background diffusion coefficient and $w_s(z,t) \leq 0\ (m/s)$ is a sinking velocity.

The BFM benthic state variables involved in the definition of the benthic-pelagic coupling are the dissolved and particulate detritus ($Q^{(1)}_{c,n,p}$ and $Q^{(6)}_{c,n,p,s}$), whose temporal rates of change are described in the BFM core manual. These benthic state variables are physically connected with the pelagic coupled processes via the definition of the sedimentary particulate organic matter flux and the definition of the bottom boundary condition for the pelagic dissolved nutrients, as described in the following section.

### 3.2.2 The surface and bottom boundary conditions

The surface boundary conditions for all the BFM pelagic LFGs and CFFs state variables $C_p$ is a "zero flux" condition:

$$\left[\left(K_M + \chi_b\right)\frac{\partial C_p}{\partial z}\right]_{z=z_s} = 0 \tag{3.2.3}$$

with the only, but very important, exception of the dissolved nutrients (phosphate, nitrate, ammonium and silicate) and dissolved gases (oxygen, $O_2$ and carbon dioxide, $CO_2$) CFFs.

Concerning the inorganic nutrients, given the mainly coastal implementation of the BFM-POM1D system, it has been adopted a surface boundary condition defining a surface flux accounting for external (river-borne) nutrient input. The condition is defined by relaxing the prognostically computed surface nutrient ($N$) concentration to a time varying observed surface value ($N^*$):

$$\left[\left(K_M + \chi_b\right)\frac{\partial N}{\partial z}\right]_{z=z_s} = \gamma\left[N^*(t) - N(z_s,t)\right] \tag{3.2.4}$$

where $\gamma$ is a user specified relaxation velocity $(m/s)$.

The surface boundary condition for $O_2$ and $CO_2$ ($G$) accounts for the ocean-atmosphere exchanges fluxes ($\Phi_{BFM(G)}$) in $mmol\left(O_2, CO_2\right) / \left(m^2 s\right)$ that are computed by a specific BFM procedure based on Wanninkkof (1992). The surface boundary condition for such BFM state variables is then:

$$\left[\left(K_M + \chi_b\right)\frac{\partial G}{\partial z}\right]_{z=z_s} = \Phi_{BFM(G)} \tag{3.2.5}$$

where $G$ indicates the $O_2$ or $CO_2$ surface concentration.

At the ocean surface the sinking vertical velocity (see sec. 3.2) is nil for all the BFM state variables:

$$w_s(0,t) = 0 \tag{3.2.6}$$

At the ocean bottom, the benthic-pelagic coupling involves the burial of pelagic particulate organic matter ($R^{(6)}_{c,n,p,s}$), and the phytoplankton functional types experiencing sinking ("diatoms", $P^{(1,)}_{c,n,p,s}$ and

"dinoflagellates", $P^{(4)}_{c,n,p}$ functional types) and the release toward the water column of dissolved inorganic nutrients. The benthic-pelagic coupling is then defined by the following bottom boundary conditions:

$$w_s(-H,t) = w_b \tag{3.2.7}$$

$$(K_H + \chi_b) \frac{\partial \left( R^{(6)}_{c,n,p,s}, P^{(1,4)}_{c,n,p,s} \right)}{\partial z} |_{z=z_b} = w_b \left[ \left( R^{(6)}_{c,n,p,s}, P^{(1,4)}_{c,n,p,s} \right) \Delta z^{-1} \right]_{z=z_b} \tag{3.2.8}$$

$$\left[ (K_H + \chi_b) \frac{\partial N}{\partial z} \right]_{z=z_b} = \frac{\mu_{Q^{(6)}_{n,p,s}}}{\Delta z |_{z=-H}} Q^{(6)}_{n,p,s} \tag{3.2.9}$$

Note that the boundary condition 3.2.9 is handled by the BFM benthic closure scheme.

## 3.3 The transformed equations

The adoption of a fractional ($\sigma$) vertical coordinate, requires recasting the governing equations into the new coordinate system. Posing $t = t^*$ and $z = \sigma H$ (refer to eq. 3.1.1). The relations (for a generic pelagic physical or biogeochemical state variable $C_p$) linking the derivatives in the old and new systems are::

$$\frac{\partial C_p}{\partial t} = \frac{\partial C_p}{\partial t^*}$$

$$\frac{\partial C_p}{\partial z} = \frac{1}{H} \frac{\partial C_p}{\partial \sigma}$$

The complete set of the physical biological coupled governing equations are then recasted in the new coordinate system as follows (the $^*$ is hereafter dropped for notational convenience):

Momentum

$$\frac{\partial (u,v)}{\partial t} \mp f(v,u) = \frac{1}{H^2} \frac{\partial}{\partial \sigma} (K_M + \chi_M) \frac{\partial (u,v)}{\partial \sigma} \tag{3.3.1}$$

Potential temperature

$$\frac{\partial T}{\partial t} = \frac{1}{H^2} \frac{\partial}{\partial \sigma} (K_H + \chi_T) \frac{\partial T}{\partial \sigma} + \Phi_T - \frac{1}{H} \frac{\partial R}{\partial \sigma} \tag{3.3.2}$$

Salinity:

$$\frac{\partial S}{\partial t} = \frac{1}{H^2} \frac{\partial}{\partial \sigma} (K_H + \chi_S) \frac{\partial S}{\partial \sigma} + \Phi_S \tag{3.3.3}$$

Hydrostaticity:

$$\frac{\partial p}{\partial \sigma} = -H\rho g$$

Twice the kinetic energy:

$$\frac{\partial q_2}{\partial t} = \frac{1}{H^2}\frac{\partial}{\partial \sigma}\left(K_H + \chi_H\right)\frac{\partial q_2}{\partial \sigma} + \frac{2g}{\rho_0}K_H\frac{1}{H}\frac{\partial \widetilde{\rho}}{\partial \sigma} - \frac{2q_2^3}{B_1 l} \tag{3.3.4}$$

Twice the kinetic energy times the turbulence length scale:

$$\frac{\partial q_2 l}{\partial t} = \frac{1}{H^2}\frac{\partial}{\partial \sigma}\left[\left(K_Q + \chi_Q\right)\frac{\partial q_2 l}{\partial \sigma}\right] + \frac{lE_1 g}{\rho_0}\left(K_H + \chi_H\right)\frac{1}{H}\frac{\partial \widetilde{\rho}}{\partial \sigma} - \frac{q_2^3}{B_1}\left[\left[1 - \left\{E_2\left[\frac{l}{\kappa\sigma H\left(1+\sigma\right)}\right]^2\right\}\right]\right] \tag{3.3.5}$$

Generic dissolved or particulate (non sinking) BFM pelagic state variables:

$$\frac{\partial C_p}{\partial t} = \frac{1}{H^2}\frac{\partial}{\partial \sigma}\left(K_H + \chi_S\right)\frac{\partial C_p}{\partial \sigma} + \frac{\partial C_p}{\partial t}\,|_{bgc}$$

Generic particulate (sinking) BFM pelagic state variables:

$$\frac{\partial C_p}{\partial t} = \frac{1}{H^2}\frac{\partial}{\partial \sigma}\left[\left(K_H + \chi_b\right)\frac{\partial C_p}{\partial \sigma}\right] + \frac{1}{H}\frac{\partial\left(w_s C_p\right)}{\partial \sigma} + \frac{\partial C_p}{\partial t}\,|_{bgc}$$

## 3.4 Information flow and numerical integration

The time evolution of physical variables is managed by the POM1D component of the modeling system and transferred to BFM. The information flow occurring between the model components of the system is schematized in Fig. 3.2. The physical variables are used to compute the diffusion and reaction terms in eq. 3.2.2 and then combined to obtain the forward in time biogeochemical states. Red arrows indicate the external data (forcing functions) that are needed by the model in both the prognostic and diagnostic mode; blue arrows indicate additional data to be provided for the prognostic mode, while the green arrow indicates the prescribed, time varying, temperature and salinity vertical profiles, required by the diagnostic mode. The numerical integration method and scheme, used to compute the forward in time solution of all the BFM-POM1D state variables, is the same. The sensitivity of the BFM-POM1D to coupling technique and to integration schemes has been tested by Butenschön et al. (2012), and it was found that the source splitting (SoS) technique is more accurate. POM1D uses, for the active tracers, a time integration method based on SoS with a leapfrog scheme. The BFM-POM1D system carries out the final integration of the non-conservative tracers adopting the same coupling technique and numerical scheme. The leapfrog numerical scheme is adopted also for the integration in time of the benthic particulate organic matter ($Q^{(1,6)}_{c,n,p,s}$).

The numerical solution of all the BFM-POM1D state variables is carried out in two steps according to the SoS technique, involving an explicit and an implicit integration. The explicit leapfrog generate

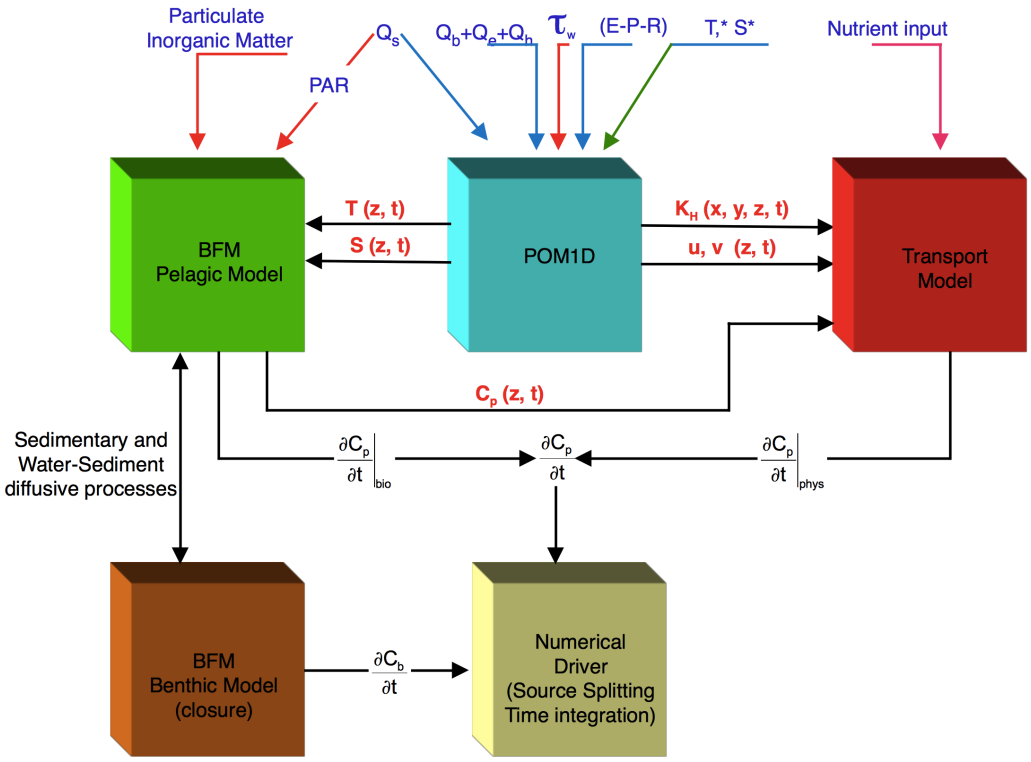


Figure 3.2: Scheme of the information flow between the ocean model and the biogeochemical state variables. The red forcing arrows refer to the use in both the prognostic and diagnostic mode. The blue arrows refer to the use in the prognostic mode only while the green arrows refer to the use in diagnostic mode only. Numerical driver is a generic name for the source splitting solver used to advance in time the coupled solution of the BFM state variables, while the forward in time solution of the physical state variables in embedded in POM1D

an intermediate solution. It might not be needed and, when needed, involves different equation terms in the specific equation. The characteristics of the (if needed) explicit leapfrog integration are given below by reporting the equation term in discrete form (refer to fig. 3.1). In the following $n$ is the time index, $k$ the vertical space index (from sea surface to bottom), $\Delta Z_k$ and $\Delta ZZ_k$ the vertical staggered grid fractional spacing.

### Momentum

The intermediate velocity values $(\widetilde{u},\widetilde{v})$ are obtained by explicitly integrating the Coriolis terms in eq. 3.3.1:

$$\frac{(\widetilde{u},\widetilde{v})^{k+\frac{1}{2}}-(u,v)_{n-1}^{k+\frac{1}{2}}}{2\Delta t}=\pm f(v,u)_{n}^{k+\frac{1}{2}} \tag{3.4.1}$$

### Conservative scalar properties ($T$, $S$, $q_2$, $q_2l$)

No explicit integration is needed for $q_2$ and $q_2l$ needed. Then the intermediate values $(\widetilde{q_2})$ and $\widetilde{q_2l}$ are given by:

$$(\widetilde{q_2},\widetilde{q_2l})^{k+\frac{1}{2}}=(q_2,q_2l)_{n-1}^{k+\frac{1}{2}}$$

For temperature and salinity the intermediate values $\widetilde{T}$ and $\tilde{S}$ are given (for the prognostic mode only) by the forward in time integration of the lateral advective fluxes. The intermediate values are then given by:

$$\frac{(\widetilde{T},\widetilde{S})^{k+\frac{1}{2}}-(T,S)_{n-1}^{k+\frac{1}{2}}}{2\Delta t}=(\Phi_T,\Phi_S)_{n}^{k+\frac{1}{2}} \tag{3.4.2}$$

### Dissolved and particulate (non sinking) pelagic BFM state variables

The intermediate value $(\widetilde{C_p})$ is obtained by explicitly integrating the biogeochemical temporal rate of change:

$$\frac{\widetilde{C_p}^{k+\frac{1}{2}}-C_{p_{n-1}}^{k+\frac{1}{2}}}{2\Delta t}=\frac{\Delta C_{p_n}^{k+\frac{1}{2}}}{\Delta t}\mid_{bgc} \tag{3.4.3}$$

### Particulate (sinking) pelagic BFM state variables

The intermediate value ($\widetilde{C_p}$) is obtained by explicitly integrating the biogeochemical temporal rate of change and the advective (sinking) vertical flux, computed by applying an upwind discretisation:

$$\frac{\widetilde{C}_p^{k+\frac{1}{2}} - C_{p_{n-1}}^{k+\frac{1}{2}}}{2\Delta t} = \frac{1}{H}\frac{w_{s_n}^{k+1}\cdot C_{p_n}^{k+\frac{1}{2}} - w_{s_n}^{k}\cdot C_{p_n}^{k-\frac{1}{2}}}{\Delta Z^k} + \frac{\Delta C_{p_n}^k}{\Delta t}\mid_{bgc} \tag{3.4.4}$$

All the intermediate values, above hereafter $\widetilde{I} \equiv (\widetilde{u},\widetilde{v},\widetilde{T},\widetilde{S},\widetilde{q_2},\widetilde{q_2 l},\widetilde{C_p})$, are then passed to an implicit Euler backward solver operating on $2\Delta t$ for the vertical diffusion:

$$\frac{C_{n+1}^{k+\frac{1}{2}} - \widetilde{I}^{k+\frac{1}{2}}}{2\Delta t} = \frac{1}{H^2\Delta Z^k}\left[(K_I+\chi_I)_{n+1}^k\cdot\left(\frac{C_{n+1}^{k-\frac{1}{2}} - C_{n+1}^{k+\frac{1}{2}}}{\Delta ZZ^{k-1}}\right)\right] - \left[(K_I+\chi_I)_{n+1}^{k+1}\cdot\left(\frac{C_{n+1}^{k+\frac{1}{2}} - C_{n+1}^{k+\frac{3}{2}}}{\Delta ZZ^{k+1}}\right)\right] \tag{3.4.5}$$

where $C \equiv (u,v,,T,S,q_2,q2l,C_{p)})$ $K_I \equiv (K_M,K_H,K_Q)$ and$\chi_I \equiv (\chi_M,\chi_T,\chi_S,\chi_Q,\chi_B)$
Note that:

- when computing potential temperature, eq. 3.4.5 must be added of the term (in discrete form) accounting for solar radiation penetration, $(\partial R/\partial\sigma)H^{-1}$, in eq. 3.3.2.
- when computing $q_2$ and $q_2 l$, eq. 3.4.5 must be added of the second and third terms (in discrete form) appearing in the right hand side of eqs, 3.3.4 and 3.3.5.

The discretisation and the integration of the equations accounting for the benthic-pelagic coupling is made trough an explicit leapfrog scheme. The equations discretisation is straightforward and, as an example, only the discrete form of benthic organic particulate detritus is given:

$$\frac{Q_{c,n,p,s_{n+1}}^{(6)} - Q_{c,n,p,s_{n-1}}^{(6)}}{2\Delta t} = -w_b\left(R_{c,n,p,s_n}^{(6)} + P_{c,n,p,s_n}^{(1,4)}\right)\mid_{z=-H} - \mu_{Q_{c,n,p,s}^6} Q_{c,n,p,s_n}^{(6)}\mid_{z=-H}$$

It is well known that in the leapfrog numerical integration scheme the solutions at odd and even time steps can diverge slowly. This time splitting is removed by a weak filter (Asselin, 1972), applied to the pelagic and benthic (physical and biogeochemical) state variables, where the solution ($C_{n_{smoothed}}$) is smoothed at each time step according to;

$$C_{n_{smoothed}}^k = C_n^k + 0.5\upsilon\left(C_{n+1}^k - 2C_n^k + C_{n-1}^k\right) \tag{3.4.6}$$

where $C_{n_{smoothed}}^k$ is the smoothed (and final) solution and $\upsilon$ is a smoothing parameter (frequently set at 0.05).

Finally the time sequence is reset:
$C_{n-1}^k = C_{n_{smoothed}}$
$C_n^k = C_{n+1}^k$
and the iteration cycle starts anew.

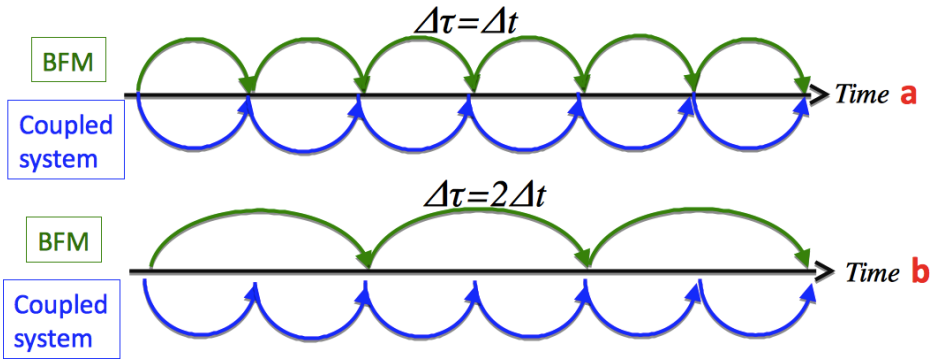


Figure 3.3: Source splitting coupling. Sinchronous coupling. b: Asynchronous coupling for the case $\Delta\tau = 2\Delta t$

The current implementation of the BFM-POM1D modeling system computes the biogeochemical rate of change with a synchronous time resolution with respect to the rates depending on physical processes. However, the coupling technique (as schematised in fig. 3.3) allows the use of a coarser resolution for the biogeochemical processes (Butenschön et al., 2012) at a time step $\Delta\tau = m\Delta t$ (with $m = 1,2,3.....$).

# 4 The computer code

The BFM site repository allows for the download of the complete BFM-POM1D computer code. Installation and code generation instruction are provided in the BFM manual. In the following `$BFMDIR` indicate the system "root" directory.

The BFM-POM1D couling interface is located in the directory `$BFMDIR/src/pom` that contains the `program main` code driving the system (`main_pombfm1d.F90`),the POM1D subroutines and modules or ocean physics (`$BFMDIR/src/pom/phys`), and the soubroutines operating the coupling (`$BFMDIR/pom/coupling`). An example of the namelists needed for the modeling system parameterisation is provided in the BFM-POM1D test case.

The complete listing of the programs, routines and modules contained in the `phys` and the `coupling` directories is given is tab. 4.1, along with the directory location and the corresponding filename.

In the following a description of the main characteristics of the `program main` code, its associated `modules`, and of all the `subroutines` is provided, with the aim to facilitate the implementation and use of the coupled modeling system.

| Name | Directory ($BFMDIR/src/pom) | Filename |
|---|---|---|
| Program MAIN | | main_pombfm1d.F90 |
| **Modules** | | |
| POM | phys | POMModule.F90 |
| CPL_VARIABLES | coupling | pom_cpl_variables.F90 |
| Forcing | coupling | pom_forcing.F90 |
| **Subroutines** | | |
| ADVERTE | phys | pom_vert_integration.F90 |
| CALCDEPTH | phys | calcdepth.F90 |
| DENS | phys | dens.F90 |
| pom_env_forcing_1d | coupling | pom_env_forcing_1d.F90 |
| FORCING_MANAGER | coupling | pom_forcing.F90 |
| load_restart | coupling | pom_restart.F90 |
| get_init_TS_IC | coupling | pom_get_init_TS_IC.F90 |
| scheme_benthic_lf1d | coupling | pom_scheme_benthic_lf1d.F90 |
| opendat | phys | opendat.F90 |
| pom_bfm_1d | coupling | pom_bfm_1d.F90 |
| pom_dia_bfm | coupling | pom_dia_bfm.F90 |
| pom_ini_bfm | coupling | pom_ini_bfm_1d.F90 |
| pom_to_bfm | phys | pom_to_bfm.F90 |
| PROFQ | phys | profq1d.F90 |
| PROF_TRACERS | phys | profTS.F90 |
| PROFUV | phys | profu.F90 |
| save_restart | coupling | pom_restart.F90 |
| Routine vert_integration | coupling | pom_vert_integration.F90 |

Table 4.1: Listing of all the programs, subroutines and modules contained in BFM-POM1D interface, the directory pathway from the reference $BFMDIR/src/pom and the corresponding filenames.

## 4.1 Program MAIN (main_pombfm1d.F90)

The main program of the BFM-POM1D coupled system has been structured around the basis the POM1D main program, to which have been added the calls to specific subroutines handling the BFM initial condition definition (or restart reading) and the execution of the BFM core.

A flow chart illustrating the structure of the main program is displayed in fig 4.1. The blue boxes indicate model branching points or loop structure, while red boxes refer to specific routines called within the main body. Black/grey boxes indicate non modular sections of code directly inserted into the main body.

The main program consist of two logical parts: the first part is the system initialisation procedure, where the model setup and the parameters relative to the specific problem treated are prepared. The second part is the time stepping loop, where the forward in time numerical integration is carried out via the index intt. In the following, these two logical parts are treated separately. A description of the general structure of the program with particular reference to the branching points and to the non modular code is provided, while the various subroutines are described in detail in dedicated sections of this report (sec. 4.3).

### Initialisation

The reading of the namelist params_POMBFM (sec. 4.4) provides the model with all the information needed for a general setup. Subsequently all the physical variables (contained in module POM) are given a first initialisation by setting them to zero and the vertical coordinate system is finalised on the basis of the parameters KL1, KL2 and KB (section 4.2.1) trough subroutine CALCDEPTH, sec. 4.3.2). At this stage the program computes also the value of the Coriolis parameter (COR), the number of iterations needed to carry out an IDAYS simulation (IEND) and twice the the time step DT2.

The value of the ihotst (sec.4.2.1) parameter detemines the branch toward a "cold" (from initial conditions) or a "hot" (from a restart file) start:

- If ihotst=0 ("cold" start ), subroutine get_init_TS_IC (sec. 4.3.7) is called. It handles the reading of the $T$ and $S$ initial conditions. Subsequently, the subroutine DENS (sec. 4.3.3 is called in order to provide the initial density field.

- If ihotst=1 ("hot" start) subroutine load_restart (sec. 4.3.6) is called, handling the reading of the file needed to provide the restart conditions for POM1D.

The main program flow converges towards the initialisation of the BFM state variables that is handled by subroutine pom_ini_bfm_1d (sec. 4.3.12), again on the basis of the ihotst value. This ends the initialisation part.

### Timestepping

Once terminated the setup and initialisation phase, the time marching loop starts and proceeds across a number of iterations defined by the loop intt=1, IEND. The first operation iteratively carried out by the time marching loop, is the definition of the time varying forcing and (in case of a diagnostic simulation) of the prescribed data. Such operations are handled by the dedicated subroutine Forcing_manager (sec. 4.3.5).

Immediately after subroutine PROFQ (sec. 4.3.14) is called. It contains the coding of the Mellor and Yamada (1982) 2.5 turbulence closure that outputs the vertical turbulent diffusion coefficient

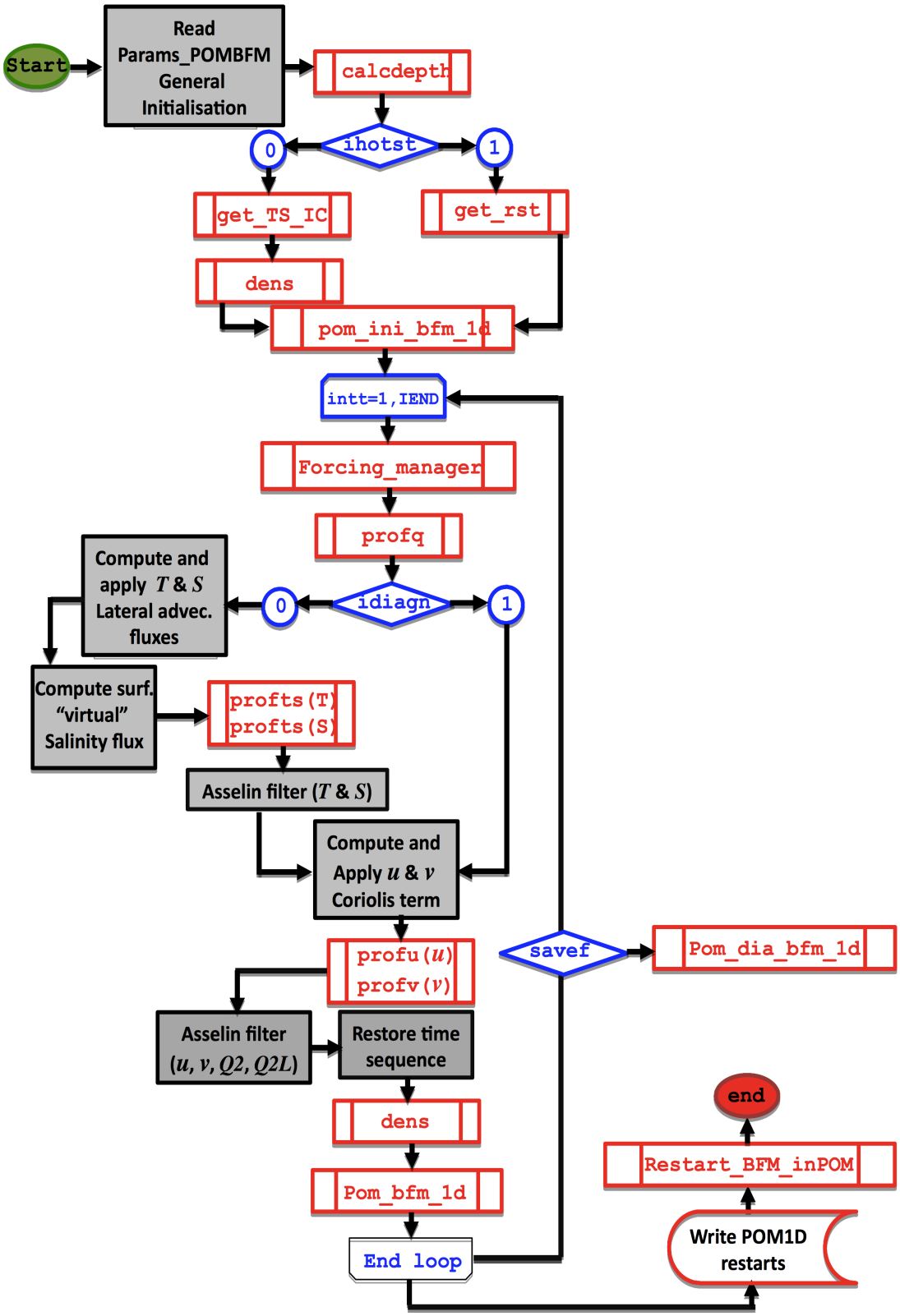


Figure 4.1: The BFM-POM1D `main` program flowchart.

$K_M K_H$ and $K_q$ to be used to compute the vertical distribution of all the state variables that are prognostically treated. The vertical $T$ and $S$ profiles are computed via the linear interpolation of the climatological data that occurs in subroutine `Forcing_manager` (sec. 4.3.5).

The computation of the Coriolis terms $fu$ and $fv$ appearing in eq. 2.2.1 is carried out. The Coriolis terms are then used to compute (via explicit leapfrog integration) the intermediate, forward in time, $\widetilde{u}$ and $\widetilde{v}$ velocity profiles (eq. 3.4.1), that are stored in the `UF` and `VF` arrays.

```
!
              UF(:) = UB(:) + DT2 * COR * V(:) ! CORIOLIS TERM AND EXPLICIT LEAPFROG
              VF(:) = VB(:) - DT2 * COR * U(:) ! CORIOLIS TERM AND EXPLICIT LEAPFROG
!
```

The temporal update of the velocity profiles is then concluded by the sequential two calls to subroutine `PROFUV` (sec. 4.3.16). After these calls the computation of the physical state variables closes with the application of the Asselin (1972) filter (eq. 3.4.6) and with the restoration of the time sequence for all the physical variables. Below it is shown the time sequence restoration for the temperature state, being identical for all the other state variables.

```
!
!     -----RESTORE TIME SEQUENCE-----
!
      TB(:) = T(:)
      T(:)  = TF(:)
!
```

Density is updated trough a call to `subroutine DENS` (sec. 4.3.3).

The call to the `subroutine Pom_bfm_1d` (sec. 4.3.10) executes BFM, as well as the physical biological coupling and the SoS time integration of the BFM prognostic variables. This last call conclude the sequence of operations contained in the time marching loop, and the cycle starts anew. However, at a `savef` frequency (see sec. 4.2.2) the routine handling the output writing is called (`subroutine pom_dia_bfm_id`, sec. 4.3.11).

## 4.2 Shared modules

### 4.2.1 Module POM (`phys/POMModule.F90`)

The description of the code must necessarily start with the `POM module`. Such module evolves from the original `include` file "`comblk.h`", defining, in many POM versions, the declaration of all the variables and parameters needed to run POM. The content of such `include` file is now transferred into this module, to which several other parameters and/or variables have been added in order to facilitate the control of the model trough values provided via namelists (see secs. 4.4 and 4.4). Here a brief description of the variables arrays and parameters (listed in alphabetical order) contained in the module is given.

`REAL(RLEN) :: ALAT`

latitude of the implementation site (*deg*). `ALAT` is used for the computation of the Coriolis Parameter `COR` (see below) in `program MAIN` (sec. 4.1). Value is provided via namelist (sec. 4.4).

`real(RLEN) :: CBC` ($c_z$ in eq.2.3.3)

The bottom drag coefficient. It is computed in the initialisation section of `program main` (sec. 4.1).

`real(RLEN), Parameter :: CBCMIN=0.0025` ($c_{z_{min}}$ in eq.2.3.3)

The minimum achievable value for the bottom drag coefficient.

`REAL(RLEN) :: COR`
Coriolis parameter $f$ ($s^{-1)}$. Computed by `program MAIN` (sec. 4.1) in initialisation section. Computation requires the `ALAT` value.

`real(RLEN), Parameter :: DAYI=ONE/SEC_PER_DAY`
One day in seconds (reciprocal. $s^{-1}$). The `REAL (RLEN), Parameter :: ONE` is contained in the BFM `module global_mem`. The `REAL(RLEN), parameter :: SEC_PER_DAY` is contained in the BFM `module constants`.

`REAL(RLEN) :: DTI`
Model time step ($s$). Value is provided trough namelist (sec. 4.4).

`real(RLEN),public,dimension(KB) :: GM,GH,SM,SH,KN,SPROD,BPROD, A, C, VH, VHP, PROD,DTEF,D, DT`
Service arrays used by the POM1D subroutines (sec. 4.3).

`real(RLEN),parameter :: GRAV=9.806`
Gravity ($m/s^2$).

`REAL(RLEN) :: H`
The bottom depth ($m$). Value is provided trough namelist (sec. 4.4).

`integer(ilong) :: IDAYS`
Length of the run ($days$). Value is provided trough namelist (sec. 4.4).

`integer(ilong) :: IEND`
Number of iterations needed to cover an "`IDAYS`" run. Computed by `program main` (sec. 4.1) in initialisation section.

`integer(ilong) ::IHOTST`
Switch for "cold"/"hot" model starts. `IHOTST=0`: "cold" start from initial conditions; `IHOTST=1`: "hot" start from a restart file reading. Value is provided trough namelist (sec. 4.4).

`integer(ilong) :: intt`
Counter of the time marching loop.

`integer(ilong),parameter :: KB=31`
Total number of vertical layers. currently defined to 31. It must be user defined HERE.

`integer(ilong) :: KL1, KL2`
Dummy arguments for the subroutine `CALCDEPTH` (sec. 4.3.2). `KL1 and KL2` are integers defining the number of surface (`KL1`) bottom (`KL2`) vertical layers having a logarithmic distribution. The number of surface/bottom logaritmic layers are (respectively) `KL1-2` and `KB-KL2-1`. Values are provided via namelist (sec. 4.4).

`real(RLEN),dimension(KB) ::KM,KH,KQ` ($K_M$, $K_H$ and $K_q$ in equations 2.2.1 to 2.2.5).
The vertical diffusion coefficients vertical profiles ($m^2/s$) for momentum (`KM`), tracers (`KH`) and kinetic energy (`KQ`). They are computed according to Mellor and Yamada (1982) in subroutine `PROFQ` (sec. 4.3.14). See also Blumberg and Mellor (1987) and Mellor (2004).

`real(RLEN),dimension(KB) :: L`
The turbulence length scale vertical profile ($m$) . The profile is prognostically computed according to Mellor and Yamada (1982) in subroutine `PROFQ(sec. 4.3.14)`. See also Blumberg and Mellor (1987) and Mellor (2004).

`integer(ilong) :: NBCT, NBCS, NBCBFM`

Flags to choose Temperature, Salinity and BFM tracers surface boundary conditions (see eqs. 2.3.4, 2.3.5, 2.3.6, 3.2.3, 3.2.4) in subroutine `PROF_TRACERS` (sec. 4.3.15). Value is provided trough namelist (sec. 4.4).

`REAL(RLEN) :: NRT` ($\gamma$ in eq. 3.2.4).

Relaxation velocity ($m/day$) for computation of surface nutrient flux (eq. 3.2.4 and sec. 4.3.18). Value is provided trough namelist (sec. 4.4). Conversion to ($m/s$) is carried out automatically by the code.

`integer(ilong) :: NTP`

Flag to choose Jerlov (1976) water type in subroutine `PROF_TRACERS` (sec. 4.3.15). Value is provided trough namelist (sec. 4.4).

`real(RLEN),dimension(KB) :: Q2F,Q2,Q2B`

Twice the turbulent kinetic energy vertical profiles (prognostically computed according to Mellor and Yamada 1982) at the three different time levels required by the leapfrog time integration scheme. `Q2B`=$Q2_{t-\Delta t}$, `Q2`=$Q2_t$, `Q2F`=$Q2_{t+\Delta t}$. The profiles are computed by subroutine `PROFQ` (sec. `4.3.14`) See also Blumberg and Mellor (1987) and Mellor (2004).

`real(RLEN),dimension(KB) :: Q2LF,Q2L,Q2LB`

The kinetic energy (twice) times the turbulence length scale vertical profiles (prognostically computed according to Mellor and Yamada 1982) at the three different time levels required by the leapfrog time integration scheme. `Q2LB`=$Q2L_{t-\Delta t}$, `Q2L`=$Q2_t$, `Q2LF`=$Q2_{t+\Delta t}$. The profiles are computed by subroutine `PROFQ` (sec.`4.3.14`). See also Blumberg and Mellor (1987) and Mellor (2004).

`real(RLEN), parameter :: RCP=4.187E6` ($\rho_0 C_p$ in eq. 2.3.4)

Water specific heat times reference density.

`real(RLEN),dimension(KB) :: RHO`

The vertical density profile ($kg/m^3$) computed from the vertical $T$ and $S$ profiles (and hydrostatic pressure) using an adaptation of the UNESCO (1981) equation of state as in Mellor (1991). Density computation is carried out in subroutine `DENS`. Note that `RHO`=$(\rho - \rho_0)\,10^{-3}$.

`real(RLEN), parameter :: RHO0=1.0e3`

A reference density ($kg/m^3$).

`real(RLEN), parameter :: RHOSEA=RHO0 + 25.0`

The seawater average density (1025 $kg/m^3$).

`real(RLEN),dimension(KB) :: SF,S,SB`

The vertical salinity (prognostically computed or diagnostically prescribed) profiles ($psu$) at the three different time levels required by the leapfrog time integration scheme. `SB`=$S_{t-\Delta t}$, `S`=$S_t$, `SF`=$S_{t+\Delta t}$. See eq. 2.2.3 and also Blumberg and Mellor (1987) and Mellor (2004).

`REAL(RLEN) :: SMOTH` ($\upsilon$ in eq. 3.4.6)

Parameter for the Asselin (1972) filter. Required when using a leapfrog integration scheme to link solutions computed at odd and even times. See also Blumberg and Mellor (1987) and Mellor (2004). Value is provided trough namelist (sec. 4.4).

`REAL(RLEN) :: SSRT` ($\alpha$ in eq.2.3.5).

Relaxation velocity ($m/day$) for the surface salinity flux. conversion to $m/s$ is carried out automatically by the code. Value is provided trough namelist (sec. 4.4).

`real(RLEN) :: SWRAD` [$(\rho_0 C_p)^{-1} Q_s$ in eq. 2.3.4].

The short wave (solar) radiation at the sea surface $(Km/s)$. Note that the POM1D code requires `SWRAD`$=-\left[(\rho_0 C_p)^{-1} Q_s\right]$ (a positive heat flux into the ocean has a negative value). When the model is run in diagnostic mode `SWRAD` is used only to provide the solar radiation to the BFM component of the modelling system. The `SWRAD` value is converted back to $W/m^2$ in subroutine `pom_to_bfm` (sec. 4.3.13).

`real(RLEN),dimension(KB) :: TF,T,TB`

The vertical potential temperature (prognostically computed or diagnostically prescribed) profiles $(C^o)$ at the three different time levels required by the leapfrog time integration scheme. `TB`$=T_{t-\Delta t}$, `T`$=T_t$, `TF`$=T_{t+\Delta t}$. See eq. 2.2.2 and also Blumberg and Mellor (1987) and Mellor (2004).

`REAL(RLEN) :: TIME`

Running time $(d)$. computed at each iteration in `program main`.

`REAL(RLEN) :: TIME0`

time at restart. `IHOTST=0: TIME0=0. IHOTST=1: TIME0` read from the restart file.

`integer(ilong) :: TRT, SRT` ($\beta_T$ and $\beta_S$).

Relaxation times $(d)$ for the definition of the lateral temperature and salinity fluxes ($\beta_T$ and $\beta_S$). Conversion to $(s)$ is operated automatically by the code. Value is provided trough namelist (sec. 4.4).

`real(RLEN) :: TSURF, SSURF` ($T^*$ and $S^*$ in eq. 2.3.6).

The time varying prescribed surface temperature and salinity values.

`real(RLEN),dimension(KB) :: UF,U,UB,VF,V,VB`

The vertical horizontal velocity components profiles $(m/s)$ at the three different time levels required by the leapfrog time integration scheme. `(UB,VB)`$=(U,V)_{t-\Delta t}$, `(U,V)`$=(U,V)_t$, `(UF,VF)`$=(U,V)_{t+\Delta t}$. See eqs. 2.2.1 and also Blumberg and Mellor (1987) and Mellor (2004).

`REAL(RLEN) :: UMOL` ($\chi_{(M,L)}$ in eqs.2.2.7 and 2.2.8 respectively).

Background diffusion value $(m^2/s)$ for the velocity $(u,v)$ and the kinetic energy $(q2, q2l)$ fields. Values are provided via namelist (sec. 4.4).

`REAL(RLEN) :: UMOLT,UMOLS,UMOLBFM` ($\chi_T$ in eq.2.2.9, $\chi_S$ in eq. 2.2.9, and $\chi_B$ in eq. 3.2.2).

Background diffusion value $(m^2/s)$ for the temperature (`UMOLT`), salinity (`UMOLS`) and the BFM (`UMOLBFM`) fields . Values are provided via namelist (sec. 4.4).

`REAL(RLEN) :: upperH` ($z_h$ in eq. )

Depth $(m)$ at which the lateral temperature and salinity fluxes start to modify the vertical profiles.

`real(RLEN), Parameter :: vonkarmann = 0.4` ($\kappa$ in eq. 2.3.3)

The von Karmann constant.

`real(RLEN) :: WUBOT,WVBOT`

The bottom stress $\left(m^2/s^2\right)$ zonal (`WUBOT`) and meridional (`WVBOT`) components divided by a reference density.They are used to specify the momentum bottom boundary conditions. See eq. 2.3.2. Computed in subroutine `PROFUV (sec. 4.3.16)`. See also Blumberg and Mellor (1987) and Mellor (2004)

`real(RLEN) :: WUSURF,WVSURF`

The wind stress $\left(m^2/s^2\right)$ zonal (`WUSURF`) and meridional (`WVSURF`) components divided by a reference density . They are used to specify the momentum surface boundary conditions. see eq. 2.3.1. Note that the POM1D code requires (`WUSURF, WVSURF`)$=-\tau^W_{(x,y)}/\rho_0$.

`real(RLEN) :: WSSURF` ($\alpha[S^*(0,t)-S(0,t)]$ in eq. `2.3.5`)
The "virtual"surface salinity flux $(psu \cdot m/s)$. see eq. It is used in subroutine `PROF_TRACERS` (sec. 4.3.15, prognostic mode only) to specify the surface boundary condition for salinity note that `WSSURF=-(`$\alpha[S^*(0,t)-S(0,t)]$ : a salt flux into the ocean carries a negative sign.

`real(RLEN), dimension(KB):: WTADV, WSADV` ($\Phi_T$ and $\Phi_S$ in eqs. 3.3.2 and 3.3.3 respectively )
The temperature $(C^o \cdot m/s)$ and salinity $(psu \cdot m/s)$ lateral advective fluxes .

`real(RLEN) ::WTSURF` $-(\rho_0 C_p)^{-1}(Q_b+Q_e+Q_h)$ in eq. 2.3.4
The loss terms of the surface heat flux. Note that the POM1D code requires `WTSURF=`$-\left[-(\rho_0 C_p)^{-1}(Q_b+Q_e+Q_h)\right]$ (a surface heat loss carries a positive sign).

`real(RLEN),dimension(KB) :: Z,ZZ,DZ,DZZ,DZR`
The arrays containing the vertically staggered coordinate system; sec.3.1. `Z` corresponds to $\sigma$ in eq. 3.1.1; `ZZ=(Z-0.5DZ)`; $DZ_k=Z_k-Z_{k+1}$; $DZZ_k=ZZ_k-ZZ_{k+1}$; $DZR=(DZ)^{-1}$. See also Blumberg and Mellor (1987) and Mellor (2004).

`real(RLEN), Parameter :: Z0B=0.01` ($z_0$ in eq. 2.3.3)
The bottom roughness length $(m)$.

### 4.2.2 Module CPL_VARIABLES (`coupling/pom_cpl_variables.F90`)

This module (contained in the directory `$BFMDIR/pom/coupling` with the filename `pom_cpl_variables.F90`) contains additional parameters, arrays and scalars necessary to accomplish the online direct coupling between POM1D and BFM. Also for this module a brief description of the content is given.

`real(RLEN),public,dimension(KB-1) :: ISM`
The time varying prescribed inorganic suspended matter concentration $\left(mg/m^3\right)$ vertical profile. it is computed via linear interpolation between time adjacent climatological values in subroutine `FORCING_MANAGER` (4.3.5). It is used to compute the light vertical extinction coefficient in the BFM subroutine `CalcVerticalExtinction` (see Vichi et al. (2022)).

`integer(ilong), Parameter :: SEC_IN_HOUR=3600`
Seconds in one hour.

`real(RLEN) :: PO4SURF,NO3SURF,NH4SURF,SIO4SURF` ($N^*$ in eq. 3.2.4)
The surface nutrient (phosphate, nitrate, ammonia, silicate respectively) prescribed concentrations in $mmol\,(P,N,Si)\,/m^3$ needed to define the surface boundary condition for nutrients .

`integer(ilong) :: savef`
Frequency $(days)$ of the output averaging and writing. Selected variables will be averaged and written over a `savef` ( sec. 4.4) timespan.
For output averaging and writing at daily frequency set: `savef=1`
For output averaging and writing at monthly frequency set: `savef=30`

`character(200) :: wind_input, surfaceS_input, radiance_input, ism_input, Sal_input, Temp_input, Sprofile_input, Tprofile_input, heat_input, surfNut_input, read_restart`
These character variables contains the pathways to the initial condition and forcing data input. their content is provided via namelist (sec. 4.3).

### 4.2.3 Module Forcing (coupling/pom_forcing.F90)

This module contains all the definitions of arrays and scalars needed to read (and prepare for the time linear interpolation) the data needed to define the physical and biogeochemical model forcing. The module contains also the subroutine FORCING_MANAGER (sec 4.3.5), that manages the time dependent data and carries out the time linear interpolation.

The scalars and arrays contained in the module are the following:

```
INTEGER(ilong),SAVE ::  ICOUNTF, IFCHGE, IFINT
```

Counters for the linear interpolations. ICOUNTF counts the data read statements. IFINT counts the model iterations. When IFINT=IFCHGE, new data need to be read.

```
real(RLEN),public,dimension(KB-1),SAVE ::  ISM1,ISM2
```

The, adjacent in time, vertical profiles of inorganic suspended sediment ($mg/m^3$). It contributes to the definition of the PAR (Photosynthetically Available Radiation) vertical extinction coefficients.

```
REAL (RLEN),SAVE ::  NO3_1,NO3_2
REAL (RLEN),SAVE ::  PO4_1,PO4_2
REAL (RLEN),SAVE ::  NH4_1,NH4_2
REAL (RLEN),SAVE ::  SIO4_1,SIO4_2
```

The, adjacent in time, surface concentration values of nitrate (NO3_# in $mmolN/m^3$), phosphate (PO4_# in $mmolP/m^3$), ammonium (NH4_#, in $mmolN/m^3$) and silicate (SIO4_# in $mmolSi/m^3$). They contribute to the definition of the nutrients surface boundary condition.

```
REAL(RLEN),SAVE ::  RATIOF
```

Time linear interpolator. RATIOF=IFINT/IFCHGE

```
REAL(RLEN),SAVE ::  SWRAD1,SWRAD2
```

The, adjacent in time, solar radiation data ($Q_s$ in $W/m^2$). These data provide the total solar radiation (to be converted into Photosynthetically Available Radiation, PAR) to the BFM code describing primary production.

```
real(RLEN),public,dimension(KB),SAVE ::  TCLIM1,TCLIM2
real(RLEN),public,dimension(KB),SAVE ::  SCLIM1,SCLIM2
```

The, adjacent in time, vertical temperature (TCLIM# in $^oC$) and salinity (SCLIM# in *psu*) profiles. They provide the full $T$ and $S$ profiles used by BFM-POM1D.

```
REAL (RLEN),SAVE ::  WSU1,WSU2,WSV1,WSV2
```

The, adjacent in time zonal (WSU#) and meridional (WSV#) components of the wind stress ($\tau^W_{(x,y)}$ in $N/m^2$). These data are used in both modes.

## 4.3 Subroutines for physics and coupling

This section describes in detail all the subroutines and modules of the BFM-POM1D modeling system. as listed in Tab. 4.1. The subroutines are alphabetically ordered.

### 4.3.1 ADVERTE (coupling/pom_vert_integration.F90)

This subroutine is called by subroutine vert_integration (sec. 4.3.18) and compute the sinking rate of change for the BFM particulate pelagic state variables (phytoplankton and particulate organic matter) that are subject to downward vertical advection (the second term in the left hand side of eq. 3.2.2):

$$\frac{\partial C_p}{\partial t}\,|^{sinking} = \frac{\partial w_s C_p}{\partial z} \tag{4.3.1}$$

The subroutine dummy arguments are:
F: The sinking BFM state variable
FDWDT: The output sinking flux
W: The sinking velocity
The discrete form of eq. 4.3.1 corresponds to the first term in the left hand side of eq. 3.4.4
The surface and bottom boundary conditions are eqs. 3.2.6 and 3.2.7 respectively. The phytoplankton sinking velocity is time and space varying and is computed in the BFM subroutine handling the phytoplankton dynamics (subroutine PhytoDynamics; see Vichi et al., 2022). The particulate detritus constant sinking velocity is user defined trough the variable p_rR6m, whose value is set trough the BFM namelist PelGlobal_parameters. The burial velocity $w_b$ of eq. 3.2.7 for phytoplankton and particulate detritus are defined by the value of the variables p_burvel_PI (phytoplankton) and p_burvel_R6 (particulate detritus) that can be set by the user in the BFM namelist Settling_parameters.

The particulate material exiting the lower water column trough the water sediment interface flux $(w_b C_p)_{z=-H}$ enters the benthic compartment trough eq. 3.2.8.

### 4.3.2 CALCDEPTH (phys/calcdepth.F90)

This subroutine established the vertical coordinate system (sec. 3.1). The subroutine is called only once, when setting up the model, at the start of the main program (sec. 4.1) Subroutine input (KB, KL1, KL2) and output (Z, ZZ, DZ, DZZ), are handled by the module POM (sec. 4.2.1).

### 4.3.3 DENS (phys/dens.F90)

The subroutine compute the "in situ" density using the UNESCO (1981) equation of state, as adapted by Mellor (1991). The "in situ" density is determined as a function of salinity, potential temperature and pressure; the latter is approximated by the hydrostatic relation and constant density. The actual density is normalized on $10^3 kg/m^3$. Since in subroutine PROFQ (sec. 4.3.14) only gradients are needed, the density value used is $RHO = (\rho - \rho_0)10^{-3}$ in order to reduce round-off error. Subroutine input (T, S, ZZ, H, RHO0) and output (RHO) are handled by module POM. (sec 4.2.1).

### 4.3.4 pom_env_forcing_1d (coupling/pom_env_forcing_1d.F90)

This subroutine is called from subroutine pom_bfm_1d (sec. 4.3.10) and handles the calling of all the subroutines that provides BFM with all the needed information about the physical environment:

subroutine pom_to_bfm (sec. 4.3.13): Transfers the POM1D state variables into the correspondent BFM's.

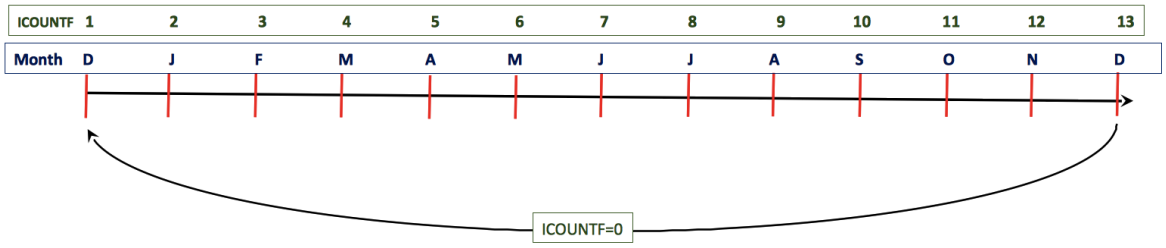


Figure 4.2: Schematic of the forcing files structure and perpetual forcing timeline. ICOUNTF is the model counter for the data readings. It is reset each time the end of file is reached

`subroutine CalcVerticalExtinction` . It is a BFM subroutine, see Vichi et al. (2022). Computes the $z$ dependent light vertical extinction coefficients.

`subroutine CalcLightDistribution`. It is a BFM subroutine, see Vichi et al. (2022). It defines the irradiance vertical profile.

### 4.3.5 FORCING_MANAGER (`coupling/pom_forcing.F90`)

This subroutine (called by `program main`, sec. 4.1) handles the forcing data reading and interpolation. In its current formulation the subroutine is structured to accomodate perpetual year monthly forcing time series. A more general version has to be provided. The data currently handled are time series of monthly averaged data (see the description of subroutine `opendat`, sec. 4.3.9). The time series of the monthly averaged data is organised as follows: DEC - JAN - FEB - MAR - APR - MAY - JUN - JUL - AUG- SEP - OCT - NOV - DEC - JAN

fig. 4.2 provides a schematic of the perpetual forcing timeline (`ICOUNTF` is the model counter for data readings and it is reset each time the end of file is reached).

The subroutine is essentially structured in three logical sections:

- initialisation and first forcing data readings
- linear time interpolation of the forcing data
- time update of the forcing data

## Initialisation and first data readings

This section is entirely embedded into the IF statement:

```
if(intt=INT(ONE)) then
          .
          .
          .
          .
endif
```

where intt is the iteration index of the time marching loop (see sec. 4.1) counting the model iterations. The IF statement ensures that the embedded statements are executed only once at the first model iteration. The following description is based on the implementation carried out in Mussap et al. (2016; 2017) and Mussap and Zavatarelli (2017). Obviously this section is open to custom modifications coeherent with the data structure that other users might have at hand.

The section start with the call to the subroutine opendat (sec. 4.3.9), that operates the opening of the data files. Subsequently counters and parameters needed for the time linerar interpolation are initialised and defined:.

ICOUNTF counts the data readings and it is initialised to 1.

IFCHGE carries the number of model iterations needed to cover a 30 days period.

IFINT progressively count the iterations. It is reset to zero in the interpolation section, when IFINT=IFCHGE. It is initialized at (IFCHGE/2)-1 because of the mid-month centered value assumption.

Afterward all the forcing data undergoes a double reading, allowing the system to load the two (monthly) data adjacent in time that are required to carry out the time linear interpolation. The update of the ICOUNTF counter closes the initialization and first reading section.

## Linear time interpolation of the forcing data

The time linear interpolation is carried out according to the general formula:

$$D_m = D_{tf} + \theta_f (D_{tf+1} - D_{tf}) \tag{4.3.2}$$

Where $D_m$ is the interpolated forcing data to be provided to the model, $D_{tf}$ and $D_{tf+1}$ are the original forcing data adjacent in time and $0 \leq \theta_f \leq 1$ is the linear interpolator. $t_f$ indicates the temporal sequence in the forcing data series.

The linear time interpolation section carries out the update of the IFINT counter and the computation of the interpolator RATIOF ($\theta_f$ in eq. 4.3.2).

```
-----UPDATE INTERPOLATION COUNTER-----
!
      IFINT  = IFINT + INT(ONE)
!
-----UPDATE INTERPOLATOR-----
!
      RATIOF = FLOAT(IFINT)/FLOAT(IFCHGE)
```

The linear time interpolation is carried out for each forcing scalar and array (wind stress, solar radiation, heat flux loss term, surface nutrients concentration, inorganic suspended matter, temperature and salinity profiles) as follows:

Scalars (only the procedure for the zonal wind stress components is shown, being identical for all the other forcing terms):

```
      WUSURF = WSU1 + RATIOF * (WSU2-WSU1)
```

where, with respect to eq. 4.3.2 WUSURF corresponds to $D_m$, WSU1 and WSU2 correspond respectively to $D_{tf}$ and $D_{tf+1}$.

Arrays: The interpolated temperature and salinity profiles are initially loaded into the TSTAR and SSTAR arrays:

```
!     -----INTERPOLATE T&S PROFILES-----
!
!       TSTAR(:) = TCLIM1(:) + RATIOF * (TCLIM2(:)-TCLIM1(:))
        SSTAR(:) = SCLIM1(:) + RATIOF * (SCLIM2(:)-SCLIM1(:))
!
```

The inorganic suspended matter profiles (ISM) are obtained with an interpolation procedure identical to the one described for temperature and salinity.

## Time update of the forcing data

The last logical section of subroutine FORCING_MANAGER handles the time dependent update (reading) of the forcing data. A data update is required each time that IFINT=IFCHGE. This activate all the statements that are embedded within the IF statement:

```
!
!     -----BEGIN DATA UPDATE SECTION-----
!
     IF (IFINT==IFCHGE) THEN
              .
              .
              .
              .
     ENDIF
```

The update procedure require an update of the data reading counter (ICOUNTF) and a reset of the iteration counter (IFINT):

```
!
!       -----....UPDATE MONTH COUNTER....-----
!
        ICOUNTF = ICOUNTF + 1
!
!       -----....RESET INTERPOLATION COUNTER....-----
!
        IFINT   = INT(ZERO)
!
```

after that, a shift in the forcing data is operated: $D_{tf+1}$ is transferred into $D_t$ (refer to eq. 4.3.2). In the following is shown only the corresponding code procedure for the zonal wind stress and temperature, the procedure for other scalars and arrays being identical:

```
!
!        -----....SHIFT THE MONTHLY DATA....----- !

         WSU1       = WSU2
!
         TCLIM1(:)  = TCLIM2(:)
!
```

A particular case occurs when the end of the forcing files is reached. In the current configuration this occurs when ICOUNTF>13. This condition activate the statements embedded into the IF statement:

```
!
!            -----IF 12 MONTHS HAVE GONE.... -----
!
             IF (ICOUNTF.GT.13) THEN
                       .
                       .
                       .
                       .
             ENDIF
```

the data reading counter is reset:

```
!
!            -----RESTART THE READING SEQUENCE-----
!
             ICOUNTF = 2
!
```

and the $D_{tf}$ data are read from the beginning of the file:

```
!
             READ (11,REC=1) WSU1,WSV1
!
             DO K = 1,KB
                READ (15,REC=K) TCLIM1(K)
             END DO
!
```

and finally the $D_{tf+1}$ scalars and arrays are updated:

```
!
!         -----READ FOLLOWING MONTH-----
!
          READ (11,REC=ICOUNTF) WSU2,WSV2
!
          DO K = 1,KB
             READ (15,REC=(ICOUNTF-1)*KB+K) TCLIM2(K)
          END DO
!
```

It is reminded that every time the $D_{tf}$ and the $D_{tf+1}$ wind-stress and surface data are updated the conversion to the POM units is operated (change of sign included) e.g.:

```
!
!          -----WIND STRESS CONVERTED TO POM UNITS (N/m2-->m2/s2)-----
!
           WSU1 = -WSU1/RHO0
           WSV1 = -WSV1/RHO0

!
!          -----HEAT FLUX CONVERTED TO POM UNITS(W/m2-->deg.C*m/s)-----
!
           SWRAD1  = -SWRAD1/rcp
           SWRAD2  = -SWRAD2/rcp
!
```

### 4.3.6 load_restart (coupling/pom_restart.F90)

This subroutine is called only if `IHOTST=1` (secs. 4.2.1 and 4.1) and handles the opening and reading of the restart file for the POM component of the BFM-POM1D modeling system. The procedure require the opening and reading of the `pom_input` namelist (sec. 4.4), that carries the path to the restart file. The open statement currently contained in the routine is corresponding to the unformatted restart writing contained in `program main` (sec. 4.1). Possible problems with the namelist opening and reading are flagged by a printed message operated by the BFM subroutine `error_msg_prn`.

### 4.3.7 get_init_TS_IC (coupling/pom_get_init_TS_IC.F90)

When `IHOTST=0` (see secs. 4.2.1), `program main` (sec. 4.1) calls this subroutine, that provides the initial temperature and salinity initial values needed for a "cold"start. As for subroutine `load_restart` (sec. 4.3.6) the path to the data files to be read is provided by the opening and reading of the `pom_input` namelist (sec. 4.4); after that, the files containing the initial $T$ and $S$ values are opened and read. The current `READ` procedure embedded into the routine refers to the model implementation used by Mussap et al. (2016; 2017) and Mussap and Zavatarelli (2017), but obviously the initial conditions file reading is open to user modification. The subroutine operates only the loading of the

initial Temperature and Salinity data in the state variables T, TB, S, and SB (see sec. 4.2.1) in order to comply with the model "cold" start. Problems in the namelist opening and reading are flagged by the BFM subroutine error_msg_prn.

### 4.3.8 SCHEME_BENTHIC_LF1D (coupling/pom_scheme_benthic_lf1d.F90)

This subroutine is called by subroutine pom_bfm_1d (sec. 4.3.10); it carries out the the time integration of the BFM benthic (scalar) state variables using a leapfrog scheme. Input and output are handled by the module POM and the BFM modules api_BFM and Mem.

The leapfrog integration is carried out according to eq. 3.4.3, where, in this case, $\widetilde{C}_p^{k+\frac{1}{2}} = \widetilde{C}_{p_{n+1}}^{k+\frac{1}{2}}$ indicates the forward in time $(t+\Delta t)$ final solution for a scalar variable (stored in the temporary variable tempo) and $C_{p_{n-1}}^{k}$ the scalar variable at time $t-\Delta t$ , that is stored into the BFM variable D2STATEB_BEN.

The computed solution is eventually "clipped" if negative values are generated (unlikely case). In this case the forward in time solution is reset to a very low value:

```
!
!    -----CLIPPING (IF NEEDED....)-----
!
    do  n = 1,NO_D2_BOX_STATES_BEN
!
            tempo(n)=max(p_small,tempo(n))
!
    end do
!
```

The Asselin filter (eq. 3.4.6) is then applied (using also the solution at time $t$, D2STATEB_BEN), and finally the time sequence is restored (see sec. 4.3.12 for the definition of the D2STATEB_BEN array).

```
!
!    -----RESTORE TIME SEQUENCE-----
!
     D2STATEB_BEN(:,1)=D2STATE_BEN(:,1)
     D2STATE_BEN(:,1)=tempo(:)
!
```

### 4.3.9 opendat (pom_opendat.F90)

This subroutine provides the opening of all the data files needed for the model setup. The file data path is provided to the routine by the reading of the namelist pom_input (sec. 4.4). In the subroutine version reported here (Mussap et al. 2016; 2017, Mussap and Zavatarelli, 2017) the open statements is referred to files that have been written in direct access mode. Oviously this is not mandatory. The structure of the input data file is open to user defined changes. Users must, however, make sure that the reading of the data in subroutines FORCING_MANAGER (sec. 4.3.5), load_restart (sec.4.3.6) and get_init_TS_IC (sec.4.3.7) is consistent with the data file access definition contained in the open statement.

### 4.3.10 pom_bfm_1d (coupling/pom_bfm_1d.F90)

The BFM-POM1D coupling is entirely handled by this subroutine (called by program main). It is structured as a simple sequential series of subroutine calls that perform:

The passage, from POM1D to BFM, of the information about the physical environment (subroutine pom_env_forcing_1d, sec. 4.3.4).

The execution of the BFM core that computes the biogeochemistry dependent rate of change of all the BFM state variables. (BFM `subroutine EcologyDynamics`. See Vichi et al., 2022).

The computation of the physics dependent (vertical diffusion and sinking) rate of change for the BFM state variables, and the forward in time integration of the pelagic BFM state Variables, with source splitting coupling method and leapfrog numerical scheme (subroutine `vert_integration`, sec. 4.3.18).

The forward in time integration of the benthic state variables with a leapfrog numerical scheme (subroutine `SCHEME_BENTHIC_LF1D`. see sec. 4.3.8).

The handling of the model output (subroutine `pom_dia_bfm`., sec. 4.3.11).

The resetting of the BFM state variables trend arrays at the end of each time iteration (BFM subroutine `ResetFluxes`. Vichi et al., 2022).

Before the handling of the model output (call to subroutine `pom_dia_bfm`) the local variable `0<TT<savef` (see secs. 4.2.1and 4.4) is updated.

The initialisation occurs at the first model iteration:

```
!
      if(first) then
!
          TT=time-time0-(dti/SEC_IN_DAY)
          out_delta=savef
          first=.false.
!
      endif
!
```

at each iteration `TT` it is incremented by a `DTI/SEC_IN_DAY` amount (The model time step converted in day fraction).

```
!
      TT = TT + dti/SEC_IN_DAY
!
```

`TT` records the time (in days) elapsed between two adjacent output writings. It is passed as dummy argument to subroutine `pom_dia_bfm` (sec. 4.3.11) and, when `TT=savef+(DTI/SEC_IN_DAY)`, the output data writing procedure is triggered.

### 4.3.11 pom_dia_bfm (coupling/pom_dia_bfm_1d.F90)

This subroutine is called by subroutine `pom_bfm_1d` (sec. 4.3.10) and handles the time averaging and saving of the model output. The time frequency of the averaging and writing is set by the value of the `savef` (sec. 4.2.1) variable (expressed in hours). Its value is defined in the namelist `params_POMBFM` (sec. 4.4). The routine only dummy argument is the variable `TT` (sec. 4.3.10). The subroutines calls the BFM subroutine `calcmean_bfm`, (Vichi et al., 2022) for output summation, averaging and writing).

### 4.3.12 pom_ini_BFM (coupling/pom_ini_bfm_1d.F90)

This subroutine is called by `program main` and ensures a BFM setup and state variable initialization consistent and coherent with the coupled 1D modelling system setup. All the requested input is managed trough modules. The BFM 1D array size is defined by the `KB` value fixed in `module POM`, sec. `4.2.1.`

It has to be noted that this routine originates from a modification of the 3D version. Therefore the size of the (originally 3D) BFM arrays is shrunk to a 1D vector along the "vertical" ($z$) dimension.

Moreover, due to the vertical grid staggering (sec. 3.1) the lowermost gridpoint (located at K=KB) is not used. Then the BFM arrays have dimension (1, 1, kb-1):

```
!
!  ----SET THE BFM ARRAYS DIMENSION-----
!
!  ***************************************************************************
!  ***************************************************************************
!  **                                                                       **
!  ** SINCE THIS IS A 1D IMPLEMENTATION, THE SIZE OF THE ARRAYS IS SHRUNK    **
!  ** TO A 1D VECTOR  ALONG the "VERTICAL" DIMENSION. (1, 1, KB-1)           **
!  **                                                                       **
!  ***************************************************************************
!  ***************************************************************************
!
     NO_BOXES=KB-1
     NO_BOXES_X=1
     NO_BOXES_Y=1
     NO_BOXES_Z=NO_BOXES
     NO_BOXES_XY=1
!
```

Detailed information on the BFM scalars defining the BFM arrays dimension can be found in Vichi et al. (2022). However, below some basic information are repeated below:

NO_BOXES: Total number of grid points where coupled system solution must be computed

NO_BOXES_Z: vertical dimension of the arrays (in a 1D setup NO_BOXES_Z=NO_BOXES)

NO_BOXES_XY: Number of surface or bottom gridpoints (in a 1D setup NO_BOXES_XY=1 by definition).

The routine then define and initialize all the masks needed for the computation (in a 1D setup this is almost redundant, however), defines the total number of variables to be computed and initialize also the netcdf output files, trough the BFM (Vichi et al., 2022) subroutines set_var_info_bfm, init_var_bfm and init_netcdf_bfm (all these operations do not need particular input from the user other than that defined by the namelists described above).

The BFM namelist bfm_general (Vichi et al., 2022) is read by the BFM subroutine init_bfm that also write information about the setup in the simulation logfile.

The namelist bfm_general contains also the variable bfm_init, that defines the simulation "cold" (from initial conditions) or "hot" (from a restart file) start. In the BFM-POM1D setup the leading "cold"/"hot" start flag is the parameter IHOTST, defined in the the module POM (sec. 4.2.1), whose value is set in the namelist 4.4. The bfm_init value is then overwritten with the IHOTST value:

```
!
!   ****************************************************************
!   ****************************************************************
!   **                                                            **
!   ** FIX THE FLAG INDICATING THE "COLD"/"HOT" START             **
!   **                                                            **
!   ** N.B.: IN THE BFM-POM1D SETUP THE LEADING  "COLD"/'HOT"     **
!   **       START FLAG IS IHOTST (DEFINED IN THE params_POMBFM   **
!   **       NAMELIST).                                           **
!   **       THE bfm_init VALUE READ IN BFM_GENERAL IS OVERWRITTEN **
!   **       WITH THE IHOTST VALUE                                **
!   **                                                            **
!   ****************************************************************
!   ****************************************************************
!
    bfm_init=ihotst
!
```

The thickness of each model layers is computed from the fractional vertical coordinate (DZ) and the bottom depth (H) to be stored into the BFM array Depth:

```
!
!    -----SET THE THICKNESS OF THE WATER COLUMN LAYERS-----
!
    do k = 1 , NO_BOXES_Z
       Depth(k) = dz(k)*h
```

```
    end do
!
```

and then the output averaging and writing procedure is initialised trough a call to BFM subroutine `calcmean`.

This routine also takes care of the definition of the initial conditions for the BFM state variables. Initial conditions are defined in two steps. A very generic initialisation is carried out by mean of the BFM (Vichi et al., 2022) subroutine `init_var_bfm`, where the BFM state variables are initialised at a spatially constant value. The initialisation is based on the carbon ($C$) content values that are fed to the routine by the reading of the `namelist bfm`. The initialisation is then completed by defining the nitrogen ($N$), phosphorus ($P$),and silicon ($Si$) content by assuming a Redfield et al. (1963) $C:N:P$ ratio, a Richards (1958) $Si:P$ ratio and on literature derived Chlorophyll to Carbon ($C:Chl$) ratio.

The numerical integration method of choice for the BFM-POM1D system is the leapfrog scheme (see sec. 3.4). The scheme requires to retain (at each time step $t$) the state variables solution at the previous time step $(t-\Delta t)$. For such time level, the values relative to the pelagic domain are stored in the array `D3STATEB`, while those relative to the benthic domain are stored in the array `D2STATEB_BEN`:

```
!
!  ******************************************************************
!  ******************************************************************
!  **                                                              **
!  ** ALLOCATE AND INITIALISE  THE ARRAY TO BE LOADED WITH STATE    **
!  ** VARIABLES (PELAGIC AND BENTHIC COMPUTED AT t=t-dt.            **
!  ** THIS IS NEEDED FOR THE LEAPGROG NUMERICAL INTEGRATION SCHEME  **
!  ** USED BY THE BFM-POM1D SYSTEM                                 **
!  **                                                              **
!  ******************************************************************
!  ******************************************************************
!
   allocate(D3STATEB(NO_D3_BOX_STATES,NO_BOXES)) !PELAGIC
!
!  -----ZEROING-----
!
   D3STATEB = ZERO
!
#ifdef INCLUDE_BEN
!
   allocate(D2STATEB_BEN(NO_D2_BOX_STATES_BEN,NO_BOXES_XY)) ! BENTHIC
!
!  -----ZEROING-----
!
   D2STATEB_BEN = ZERO
!
#endif
!
```

In case of a "cold" start, the initial conditions are loaded in both the arrays corresponding to the $t$ (`D3STATE, D2STATE_BEN`) and the $t-\Delta t$ (`D2STATEB, D2STATEB_BEN`) time levels. More information on the structure pf the BFM arrays are provided in Vichi et al. (2022).

Finally, the restart file is initialised and read based on the value of `bfm_init` (calls to the BFM subroutines `init_netcdf_rst_bfm` and `read_rst_bfm`.

### 4.3.13 pom_to_bfm (phys/pom_to_bfm.F90)

This subroutine is (although very simple) an important element of the BFM-POM1D coupling, since it provides the transfer of the information about the physical environment towards BFM. Information transferred (temperature, salinity, density, inorganic suspended matter, surface shortwave radiation, wind velocity, bottom stress) is used to compute the biogeochemical rate of change (eq. 3.2.1) of the BFM state variables.

Essentially the subroutine operate a simple data transfer between arrays, with the only difference of the wind velocity, that is (approximatively) computed back from the wind stress data assuming an air density values $\rho_a = 1.25$ and a drag coefficient $1.40 \cdot 10^{-3}$.

| | Origin | Destination | |
|---|---|---|---|
| Property | Array name and conversion algorithm | Array name | Units |
| Potential Temperature | TB(k) | `ETW(k)` | $^{\circ}C$ |
| Salinity | SB(k) | `ESW(k)` | $psu$ |
| Density | [RHO(k)·$10^3$]+$\rho_0$ | `ERHO(k)` | $kg/m^3$ |
| Pressure | RHO(k)·$g$·ZZ(k)·H·$10^{-5}$ | `EPR(k)` | $dbar$ |
| Inorg. Susp Matter | ISM(k) | `ESS(k)` | $mg/m^3$ |
| Radiance | -SWRAD·($\rho_0 \cdot C_p$) | `EIR(1)` | $W/m^2$ |
| Wind velocity | [(WUSURF$^2$+WVSURF$^2$)$^{0.5}$/($\rho_a \cdot C_d$)]$^{0.5}$ | `EWIND` | $m/s$ |
| Bottom Stress | (WUBOT$^2$+WVBOT$^2$)$^{0.5}$ | `ETAUB` | $N/m^2$ |

Table 4.2: Array to array transfer (and conversion algorithms) operated by `subroutine pom-to-bfm`.

The correspondence between the "origin" and "destination" arrays involved in the transfer is summarized in tab. 4.2

### 4.3.14 PROFQ (`phys/profq1D.F90`)[1]

This is the Mellor and Yamada (1982) 2.5 turbulence closure model. It is called by `program main`. Its only dummy argument is twice the time step `DTI`. The other necessary input and the model output is managed trough modules.

This subroutine first solves for the vertical part of the equations 2.2.4 and 2.2.5 for $q_2$ and $q_2 l$. The solution is based on the implicit integration scheme (see eq. 3.4.5) The numerical procedure is the same described for subroutine `PROF_TRACERS` (sec. 4.3.15). A somewhat simplified version of the level 2.5 model is used here and is discussed in Galperin et al. (1988) and Mellor (1989). The vertical diffusivities, $K_M$ and $H_H$, are defined according to:

$$K_{(M,H)} = q_2 l S_{(M,H)}$$

The coefficients, $S_M$ and $S_H$, are functions of a Richardson number given by:

$$S_H = [1 - (3A_2B_2 + 18A_1A_2)\,G_H] = A_2\left[1 - \frac{6A_1}{B_1}\right] \tag{4.3.3}$$

$$S_M = (1 - 9A_1A_2G_H) - 9S_HA_1G_H\,(2A_1 + A_2) = A_1\left(1 - 3C_1 - \frac{6A_1}{B_1}\right) \tag{4.3.4}$$

where:

$$G_H = \left(\frac{l}{q_2}\right)^2 \frac{g}{\rho_0}\left(\frac{\partial \rho}{\partial z} - \frac{1}{C_s^2}\frac{\partial p}{\partial z}\right) \tag{4.3.5}$$

[1] Most of this section is taken from Mellor (2004)

is a Richardson number. The five constants in eqs. 4.3.3 and 4.3.4 are mostly evaluated from near surface turbulence data (law-of-the-wall region) and are found (Mellor and Yamada, 1982) to be $(A_1, B_1, A_2, B_2, C_1) = (0.92, 16.6, 0.74, 10.1, 0.08)$. The stability functions limit to infinity as $G_H$ approaches the value, 0.0288, a value larger than one expects to find in nature. The quantity, $C_s^2$, in eq. 4.3.5 is the speed of sound squared. The vertical pressure gradient is obtained from the hydrostatic relation, where the density is taken as a constant value consistent with the pressure determination in `subroutine DENS` (sec. 4.3.3) i.e. $\partial p/\partial z = -\rho_0 g$.

### 4.3.15 PROF_TRACERS (coupling/pom_profTrc.F90)[2]

This is a modification of the POM subroutine PROFT and solves for temperature, salinity and for the BFM state variables. It is called by `program main` (sec. 4.1) in order to compute temperature and salinity, and by subroutine `vert_integration` (sec. 4.3.18) in order to compute the BFM state variables (in diagnostic mode, sec.2.4, the routine is used only for the computation of the BFM state variables). Its dummy arguments are:

`FF`: Property to be solved forward in time
`WFSURF`: Property surface flux (for temperature it lacks the incoming surface solar radiation)
`WFBOT`: Property bottom flux.
`SWRAD`: Incoming solar radiation (to be used only when prognostically computing temperature).
`FSURF:` A prescribed surface value.
`NBC`: Flag for definition of the surface boundary condition.
`DT2`: Twice the Time step.
`NTP`: Flag to choose the optical (Jerlov, 1976) water type.
`UMOL`: Background diffusivity.

The implicit forward in time integration follows Richtmeyer and Morton (1967). The method is used also to compute also $u$,$v$, $q_2$,$q_2l$(see secs. 4.3.14 and 4.3.16).

In order to describe the solution procedure, eq. 3.4.5 is written for temperature (the equation is then added of the radiation term appearing in eq.3.3.2). Solution for velocity components, salinity and BFM pelagic state variables can be obtained from the following by eliminating all the terms involving radiation. Note that the indexing used is consistent with fig. 3.1.

$$T_{n+1}^{k+\frac{1}{2}} - \widetilde{T}^{k+\frac{1}{2}} = \frac{2\Delta t}{H^2 \Delta Z^k}\left[(K_H+\chi_T)_{n+1}^{k}\cdot\left(\frac{T_{n+1}^{k-\frac{1}{2}} - T_{n+1}^{k+\frac{1}{2}}}{\Delta ZZ^{k-1}}\right)\right] -$$
$$-\left[(K_H+\chi_T)_{n+1}^{k+1}\cdot\left(\frac{T_{n+1}^{k+\frac{1}{2}} - T_{n+1}^{k+\frac{3}{2}}}{\Delta ZZ^{k+1}}\right)\right] - \frac{2\Delta t}{H\Delta Z^k}\left(R_{n+1}^{k-\frac{1}{2}} - R_{n+1}^{k+\frac{1}{2}}\right) \tag{4.3.6}$$

The above equation can be written as:

$$-T_{p_{n+1}}^{k+\frac{3}{2}} a^k + T_{p_{n+1}}^{k+\frac{1}{2}}\left(a^k + c^k - 1\right) - T_{n+1}^{k-\frac{1}{2}} c^k = d^k \tag{4.3.7}$$

where:

[2]Most of this section is taken from Mellor (2004)

$$a^k = -\frac{2\Delta t(K_H + \chi_T)^{k+1}_{n+1}}{H^2 \Delta Z^k \Delta ZZ^{k-1}} \tag{4.3.8}$$

$$c^k - \frac{2\Delta t(K_H + +\chi^h_T)_{n+1}}{(\Delta z^k)^2} \tag{4.3.9}$$

$$d^k = -\widetilde{T}^{k+\frac{1}{2}} + \frac{2\Delta t}{\Delta z^k}\left(RAD^{k-\frac{1}{2}}_{n+1} - RAD^{k+\frac{1}{2}}_{n+1}\right) \tag{4.3.10}$$

assuming a solution of the form:

$$T^{k+\frac{1}{2}}_{n+1} = VH_k \cdot T^{k+\frac{3}{2}}_{n+1} + VHP_k \tag{4.3.11}$$

Inserting$T^{k+\frac{3}{2}}_{n+1}$ directly from eq. 4.3.11 and $T^{k-\frac{1}{2}}_{n+1}$ ,obtained from eq. 4.3.11, into 4.3.7 and collecting coefficients of $T^{k-\frac{1}{2}}_{n+1}$ and 1 yields:

$$VH^k = \frac{a^k}{a^k + c^k(1 - VH^{k-1}) - 1} \tag{4.3.12}$$

$$VHP^k = \frac{c^k VHP^{k-1} + d^k}{a^k + c^k(1 - VH^{k-1}) - 1} \tag{4.3.13}$$

The solution is then as follows: All $a_k$, $c_k$ and $d_k$ terms are calculated from eqs. 4.3.8, 4.3.9 and 4.3.10. Surface boundary conditions, discussed below, provide $VH_1$ and $VHP_1$, all the necessary $VH_k$ and $VHP_k$ are obtained from the descending (as $k$ increases towards the bottom) recursive relations 4.3.12 and 4.3.13. Bottom boundary conditions provide $T^{kb-\frac{1}{2}}_{n+1}$ where $kb - 1/2$ is the grid point nearest the bottom. Thereafter all of the $T^{k+\frac{1}{2}}_{n+1}$ may be obtained from the ascending recursive relation 4.3.11.

## The treatment of the short wave radiation.

The short wave radiation, is specified on the base of the optical water types classification proposed by Jerlov (1976) as interpreted by Paulson and Simpson (1977). Eq. 2.2.6is then discretized as follows:

$$\texttt{RAD=SWRAD[RP}e^{\frac{Z_k H}{Ad_1}}\texttt{+(1-RP}e^{\frac{Z_k H}{Ad_2}}\texttt{)]}$$

and the corresponding code in `subroutine PROF_TRACERS` is:

| NTP | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Jerlov (1976) type | I | Ia | Ib | II | III |
| RP ($R_p$) | 0.58 | 0.62 | 0.67 | 0.70 | 0.78 |
| AD1 ($\varsigma_1$) | 0.35 | 0.60 | 1.0 | 1.5 | 1.4 |
| AD2 ($\varsigma_2$) | 23.0 | 20.0 | 17.0 | 14.0 | 7.90 |

Table 4.3: The subroutine PROF_TRACERS parameters specifying the Jerlov (1976) optical water types. Attenuations lengths AD1 and AD2 are from Paulson and Simpson (1977).

```
!
!       ******************************************************
!       ******************************************************
!       **                                                  **
!       ** PENETRATIVE RADIATION CALCULATION.               **
!       ** AT THE BOTTOM ANY                                **
!       ** UNATTENUATED IS DEPOSITED IN THE BOTTOM LAYER     **
!       **                                                  **
!       ******************************************************
!       ******************************************************
!
        RAD(:) = SWRAD * ((      RP(NTP) * EXP(Z(:)*H/AD1(NTP))              &
                         +  (ONE-RP(NTP) * EXP(Z(:)*H/AD2(NTP)))))
!
        RAD(KB)=ZERO
!
```

where the partitioning factor $RP$ and the attenuation lengths $Ad_1$ and $Ad_2$ (respectively $R$, $\varsigma_1$ and $\varsigma_2$ in eq. 2.2.6), are functions of the subroutine dummy argument NTP, that specifies the shortvawe transmission characteristics of each Jerlov (1976) optical water type as summarized in table4.3:

## The surface boundary condition

Four different surface boundary conditions can be selected by choosing the appropriate NBC parameter when calling subroutine PROF_TRACERS:

NBC=1 - Surface boundary conditions is WFSURF . Here a surface flux (external forcing) is specified. This option can be used for all the state variables of the BFM-POM1D system. However, in case of the temperature boundary condition it has to be noted that the the dummy argument WFSURF carries only the $-\left[-(\rho_0 C_p)^{-1}(Q_b+Q_e+Q_h)\right]$ components of the total heat flux eq. 2.3.4). The short wave radiation term $-(\rho_0 C_p)^{-1} Q_s$ , in the same equation, is provided to the subroutine via the dummy argument SWRAD. The composition of the total heat flux is carried out automatically by the subroutine (the vertical radiation profile RAD is also set to zero):

```
      select case (NBC)
         case (1)
!
              VH(1)  =  A(1)/(A(1)-ONE)
              VHP(1) = -DT2*(WFSURF+SWRAD)/(-DZ(1)*H) - FF(1)
              VHP(1) =  VHP(1)/(A(1)-ONE)
!
              RAD(:) = ZERO
!
```

It is recalled that the surface "virtual" salinity flux required for the salinity surface boundary condition (eq. 2.3.5) is computed in program main (sec. 4.1), while the surface flux for the BFM state variables (eqs. 3.2.3, 3.2.4 and 3.2.5) is computed in subroutine vert_integration (sec. 4.3.18).

NBC=2 - This option is valid only for the treatment of the temperature. Surface boundary condition is WSSURF with SWRAD penetrating the water column as described in the previous paragraph . The corresponding code is:

```
        case (2)
!
              VH(1)   = A(1)/(A(1)-ONE)
              VHP(1)  = DT2*(WFSURF+RAD(1)-RAD(2))/(DZ(1)*H)  -FF(1)
              VHP(1)  = VHP(1)/(A(1)-ONE)
!
```

`NBC=3` - Surface boundary condition is `FSURF` (a prescribed surface value). This option can be used for all the state variables of the BFM-POM1D system.

`NBC=4` - This option is valid only for the treatment of the temperature. Surface boundary condition is `TSURF` and `SWRAD` (a prescribed SST plus the short wave radiation penetration).

The surface boundary conditions application can be described by writing eq. 4.3.6 for for $k = 3/2$ (as an example it is again proposed the temperature equation solution. For other state variables the radiation terms must be dropped:

$$T^{\frac{3}{2}}_{n+1} - \widetilde{T}^{\frac{3}{2}} = -\frac{2\Delta t}{H^2 \Delta Z^1}\left(WFSURF_{n+1} + RAD^1_{n+1} - RAD^2_{n+1}\right) - A_1\left(T^{\frac{3}{2}}_{n+1} - T^{\frac{5}{2}}_{n+1}\right)$$

using eq 4.3.11to eliminate $T^{\frac{5}{2}}_{n+1}$ and collecting coefficients of $T^{\frac{3}{2}}_{n+1}$ and 1 yields:

$$VH^1 = \frac{A^1}{A^1 - 1}$$

$$VHP^k = \frac{2\Delta t}{H\Delta Z^1}\left(WFSURF_{n+1} + RAD^1_{n+1} - RAD^2_{n+1} - \widetilde{T}^{\frac{3}{2}}\right)\left(\frac{1}{A^1 - 1}\right)$$

Prescribing a surface value (eq. 2.3.6) is much simpler since:

$$VH^1 = 0$$

$$VHP^k = T^*$$

**The bottom boundary condition** An adiabatic bottom boundary condition can be obtained by setting the dummy argument `WFBOT=0`. A non-adiabatic bottom boundary condition (non-zero `WFBOT`, yet to be implemented) necessary for the BFM nutrients state variables (see eq. 3.2.9) yields (repeating the above):

$$T^{km-\frac{1}{2}}_{n+1} = \frac{c^{km} - VHP^{km-1} - \widetilde{T}^{km-\frac{1}{2}}_{n+1} + \left(WFBOT + RAD_{km-1} - RAD_{km}\right) 2\Delta t / \left(H\Delta Z^{km-1}\right)}{c^{km}\left(1 - VH^{km-1}\right) - 1}$$

### 4.3.16 PROFUV (phys/profuv.F90)

This subroutine is called twice from `program main` (sec. 4.1) and computes the vertical profiles of the velocity (zonal and meridional) components. The solution method applied is the one described for subroutine `PROF_TRACERS` (sec. 4.3.15). The bottom boundary condition is, however, computed according to eq. 2.3.2.

The subroutine dummy arguments are:

`DT2`: Twice the time step
`VELF`: the velocity component to be computed
`SSTRESS`: the surface stress
`BSTRESS`: the bottom stress

### 4.3.17 save_restart (coupling/pom_restart.F90)

The subroutine is called by `program main` (sec. 4.1) after the completion of the time marching loop. It handles the writing of the restart file for the BFM. The routine simply calls the BFM subroutine `save_rst_bfm` (see Vichi et al., 2022) after the conversion of the model time from days to seconds.

```
!
! -----TIME ELLAPSED (IN SECONDS) SINCE THE BEGINNING OF THE SIMULATION----
!
   localtime = time*SEC_PER_DAY
!
!-----WRITE RESTART-----
!
   call save_rst_bfm(localtime)
!
```

### 4.3.18 vert_integration (coupling/pom_vert_integration.F90)

This subroutine (called by `subroutine pom_bfm_1d`, sec. 4.3.10) is the crucial one in the BFM-POM1D coupling procedure, as it computes the forward in time solution for the BFM pelagic state variables, accounting for the variation determined by both the physical and biogeochemical processes (eq. 3.2.1). The subroutine receives the complete biogeochemical rate of change, as computed by the BFM core, and then proceeds to the time integration of eq. 3.2.2), with a SoS numerical coupling technique, involving a semi implicit leapfrog numerical scheme as described in Butenschön et al. (2012).

The computer code consists essentially of a loop iterating over all the BFM pelagic state variables:

```
   do m = 1 , NO_D3_BOX_STATES
            .
            .
            .
            .
            .
   enddo
```

for each BFM pelagic state variable, the solution computed at time $t$ and $t - \Delta t$ is respectively transferred into the storage arrays `fbio` and `ffbio` from the master BFM arrays `D3STATE` and `D3STATEB` (Vichi et al., 2022) and sec. 4.3.12):

```
!
!          -----LOAD BFM STATE VAR.-----
!
           fbio(:)  = D3STATE(m,:)
           fbbio(:) = D3STATEB(m,:)
!
```

After that, the surface boundary diffusive fluxes for the gases (oxygen and carbon dioxide) and nutrients (nitrate ammonium, phosphate, silicate) are stored into the `surflux` and `botflux` scalars:

for gases the surface boundary diffusive fluxes (eq. 3.2.5) are computed by specific BFM procedures, see sec 3.2.2 and Vichi et al. (2022), while the surface diffusive boundary flux for nutrients is computed by applying the relaxation procedure of eq. 3.2.4, involving a prescribed climatological, time varying surface nutrient concentration ($N^*$) and a user defined relaxation velocity.

At the bottom, the gases and nutrient diffusive fluxes at the water-sediment interface are are both computed in the BFM `subroutine BenthicReturn1Dynamics` according to eq. 3.2.9 and inserted into the biogeochemical rate of change trough a procedure handled by the BFM core. The variable botflux is therefore set to zero, but future version of the code will use the bottom diffusive flux as a bottom boundary condition.

```
!
!               -----OXYGEN-----
!
                case (ppO2o)
!
                      surflux = -(jsurO2o(1)/SEC_PER_DAY)
!
                      botflux = ZERO
!
!               -----PHOSPHATE----
!
                case (ppN1p)
!
                     surflux = -(PO4SURF-n1p(1))*vrelax
!
                     botflux = ZERO
!
```

All the other BFM pelagic state variables have a nil surface and bottom diffusive flux.

After that, the subroutine proceed to store, into the array `sink`, the sinking velocity ($w_s$), converted from $m/d$ to $m/s$ for the particulate organic matter and phytoplankton functional groups. It is recalled that the particulate organic matter sinking velocity is vertically constant, and is user defined trough an assignment to the BFM parameter `p_rR6m`, contained in the namelist `PelGlobal_parameters`. the phytoplankton sinking velocity (if any) is vertically variable in dependence of the nutrient stress status and depends on the BFM parameters p_res and p_esNI (see Vichi et al. (2022)). The sinking velocity at the water sediment interface is imposed to correspond to the burial velocity $w_b$ in order to comply with the bottom boundary condition 3.2.8. The $w_b$ value for (respectively) the particulate organic matter and the phytoplankton is user defined trough an assignment to the BFM parameters `p_burvel_R6` and `p_burvePI` contained in the `Settling_parameters` namelist. As an example below it is shown the full procedure for the definition of the particulate organic matter sinking velocity.

```
!
!         ****************************************
!         ****************************************
!         **                                    **
!         ** Computing sedimentation fluxes     **
!         **                                    **
!         ****************************************
!         ****************************************
!           select case ( m )
!
!               -----PARTICULATE ORGANIC MATTER-----
!
                case (ppR6c:ppR6s)
!
                      do k = 1 , KB - 1
!
                         sink(k) = -sediR6(k)/SEC_PER_DAY
!
                      end do
!
!                     ******************************************
!                     ******************************************
!                     **                                      **
!                     ** Set the  Particulate organic matter  **
```

```
!                     ** sinking velocity (burial velocity) at the **
!                     ** water-sediment interface.                  **
!                     **                                           **
!                     ***********************************************
!                     ***********************************************
!
                      sink(kb)=-p_burvel_R6/SEC_PER_DAY
                      if(ABS(sink(kb-1)).lt.ABS(sink(kb))) &
                      sink(kb) = sink(kb-1)
!
```

For all the non-sinking BFM pelagic state variables the `sink` array is initialised at zero.

The following step is the definition of the vertical advection (sinking) flux, computed according to eq. 4.3.1. This occur trough a call to the subroutine `adverte` (sec. 4.3.1). Note that this subroutine is called for all the BFM pelagic state variables (sinking and non sinking). However the `sink(:)=0` assignment for the non sinking state variables ensures that the output sinking flux (stored in array `ffbio`) is nil.

The source splitting forward in time integration (sec. 3.4) of the BFM pelagic state variables can be now performed. The first step is the execution of an explicit leapfrog integration involving the sinking (if any) and the biogeochemical rate of change. For the non sinking state variables the integration is carried out according to eq. 3.4.3, while for the sinking state variables it follows eq. 3.4.4. The intermediate value arising from this explicit integration is stored into the array `ffbio:`and passed to the `subroutine PROF_TRACERS 4.3.15` that conclude the computation of the forward in time solution, trough an implicit integration accounting for the vertical diffusion processes acting on the state variables.

The (almost) final solution (stored in the array `ffbio`) is checked for (unlikely) negative concentration values trough a "clipping" procedure:

```
!
!     -----CLIPPING ..... IF NEEDED-----
!
      do k=1, KB-1
!
            ffbio(k)=max(p_small,ffbio(k))
!
      end do
!
```

In such case the solution is set to a very small value.

The solution is then effectively finalised trough the Asselin (1972) filtering. The storage of the solutions at the relevant time levels is then stored back into the master arrays `D3STATE` and `D3STATEB`:

```
!
!    ---MIX SOLUTIONS AND RESTORE TIME SEQUENCE-----
!
      do n = 1, KB-1
!
       D3STATEB(m,n)=fbio(n)+0.5_RLEN*smoth*(ffbio(n)+ fbbio(n)-2.0_RLEN*fbio(n))
       D3STATE(m,n)=ffbio(n)
!
      end do
```

## 4.4 BFM-POM1D setup and control namelists

The `pom_bfm_settings.nml` file is a container for the two main namelists that control the BFM-POM1D setup (`Params_POMBFM`) and initial conditions and external forcings (`pom_input`).

The namelist `Params_POMBFM` is read in the `MAIN` program and provides all the values for the variables that define the code setup and control. All of them are stored into `module POM` (sec. 4.2.1) or module `service` (sec. 4.2.2). Below the namelist set for the implementation used in Mussap et al. (2016; 2017) and Mussap and Zavatarelli (2017) is reported as an example.

```
! NAMELIST params_POMBFM
!
! NAME                    [UNIT]/KIND       DESCRIPTION
! H                           [m]           Depth
! DTI                         [s]           Timestep
! ALAT                        [degrees]     Latitude
! IDIAGN                     legacy, the runs are now only diagnostic
! IDAYS                      [d]           Length of run
! SMOTH                      [-]           Parameter for Hasselin filter
! IHOTST                     [-]           1 = Restart reading
!                                          0 = Cold start
! KL1 & KL2                  [-]           Establish Surface/bottom logarithmic layers distribution
!                                          KL1: Surface logarithmic distribution. The numers of layers is KL1-2
!                                          KL2: Bottom logarithic distribution. The numers of layers is KB-KL2-1
! SAVEF                      [d]           Saving frequency (days)
! NRT                        [m/d]         Relaxation constant for nutrients at surface
! NBCT, NBCS, NBCBFM         [-]           Define surface boundary conditions for temperature, salinity & BFM tracers !
NBC = 1: Surf. B.C. is WFSURF. no radiative penetration
!                                          NBC = 2; Surf. B.C. is WFSURF+SWRAD*(1.-TR). with SWRAD*TR penetration
!                                          NBC = 3; Surf. B.C. is TSURF. no SW radiative penetration
! UMOL                       [m^2/s]       Background diffusion for momentum
! UMOLT, UMOLS,UMOLBFM       [m^2/s]       Background diffusion for T,S and BFM tracers
! NTP                        [-]           Flag for Jerlov water type
!                                          NTP =                1      2       3       4       5
!                                          JERLOV TYPE    =     I      IA      IB      II      III
! TRT,SRT                    [ ]           Relaxation time for t & S lateral advection
! upperh                     [m]           Depth at which lat eadv start
! SSRT                       [m/d]         Relaxation time for surface salinity flux
! CBCMIN                                   Minimum achievable bottom drag coeff
! ZOB                                      Bottom roughness length
!
!  &Params_POMBFM
H       =  16.,
DTI     =  100.,
ALAT    =  45.,
IDIAGN  =  1,
IDAYS   =  3600,
SMOTH   =  0.1,
IHOTST  =  0,
KL1     =  7,
KL2     =  25,
SAVEF   =  30,
NRT     =  0.2,
NBCT    =  2,
NBCS    =  1,
NBCBFM  =  1,
UMOL    =  2.e-5,
UMOLT   =  1.e-5,
UMOLS   =  1.3e-6,
UMOLBFM =  2.e-5,
NTP     =  2,
TRT     =  0,
SRT     =  1,
UPPERH  =  5.0,
SSRT    =  5.68
/
```

This namelist `pom_input` is read three times: in subroutine `FORCING_MANAGER` (sec. `4.3.5`), subroutine `get_init_TS_IC` (sec. 4.3.7) and subroutine `load_restart` (sec. 4.3.6). It provides the directory paths to all the data files (surface forcing, prescribed $T$, $S$, inorganic suspended sediment vertical profiles and surface nutrient concentration values) to be read by the coupled system. The example below is again taken from the BFM-POM1D implementation of Mussap et. al. (2016; 2017), Mussap and Zavatarelli (2017).

```
! NAMELIST: pom_input
! -----DEFINE PATH TO DATA FILES-------
!
! NAME              [UNIT]/KIND       DESCRIPTION
! wind_input          [N/m^2]         U and V components of surface wind stress
! heat_input          [W/m^2]         Total heat flux
! surfNut_input     [BFM units]       surface nutrient inputs (NO3,NH4,PO4,SIO4)
! Sprofile_input        [-]           Salinity Initial Condition
! Tprofile_input      [degC]          Water temperature Initial Condition
! ism_input             [-]           Inorganic Suspended Matter monthly climatology
! Sal_input             [-]           Salinity monthly climatology
! Temp_input            [-]           Water temperature monthly climatology
! read_restart          char          Filename of input restart file for POM
!
&pom_input
wind_input      =  '/input/windstress_mon_Mussap.da',
surfaceS_input  =  '/input/Ssurf_16m.da',
radiance_input  =  '/input/radiance_daily_Mussap.da',
ism_input       =  '/input/ISM_16m_Mussap.da',
Sal_input       =  '/input/ClimaS_MA21_16m.da',
Temp_input      =  '/input/ClimaT_MA21_16m.da',
```

```
Sprofile_input  =  '/input/Sinit_16m.da',
Tprofile_input  =  '/input/Tinit_16m.da',
heat_input      =  '/input/heatflux.da',
surfNut_input   =  '/input/NutrientsARPAOGS.da',
read_restart    =  '/input/fort.70'
/
```

# 5 Running BFM_POM1D

## 5.1 Description

The BFM_POM1D configuration is based on the work carried out by Mussap et al. (2016), Mussap and Zavatarelli (2017) and Mussap et al. (2017) and it represent the evolution of biogeochemical processes in the middle of the Gulf of Trieste (Italy). The default configuration is a 10 years long simulation of a water column located in the centre of the gulf driven by climatological forcing conditions, with the prescription of seawater temperature and salinity profiles using the diagnostic mode (see details in section 2.4). All the input files are contained in the configuration folder BFM_POM1D and the reader is referred to the papers quoted above for details, as well for the selected parameterizations and settings.

## 5.2 Execute the model

Code compilation and deployment of the model is done automatically by the script *bfm_configure.sh* (see the BFM core manual for its usage and environment setup) by using the following command

*$> ./bfm_configure.sh -gcd -p BFM_POM1D*

Once that model setup is completed go the run folder (`${BFMDIR}/run/bfm_pom1d`) and launch the executable

*$> ./bfm_pom.exe*

The model will perform a 10 year long simulation saving output data every 30 days (see also description of `Params_POMBFM` namelist).

## 5.3 Results

As illustrated in 5.1, the configuration simulates the coupled physical and biogeochemical dynamics in the gulf of Trieste with a winter peak of phytoplankton that occurs along the entire water column and a progressive deepening of biological activity during the summer stratified conditions.

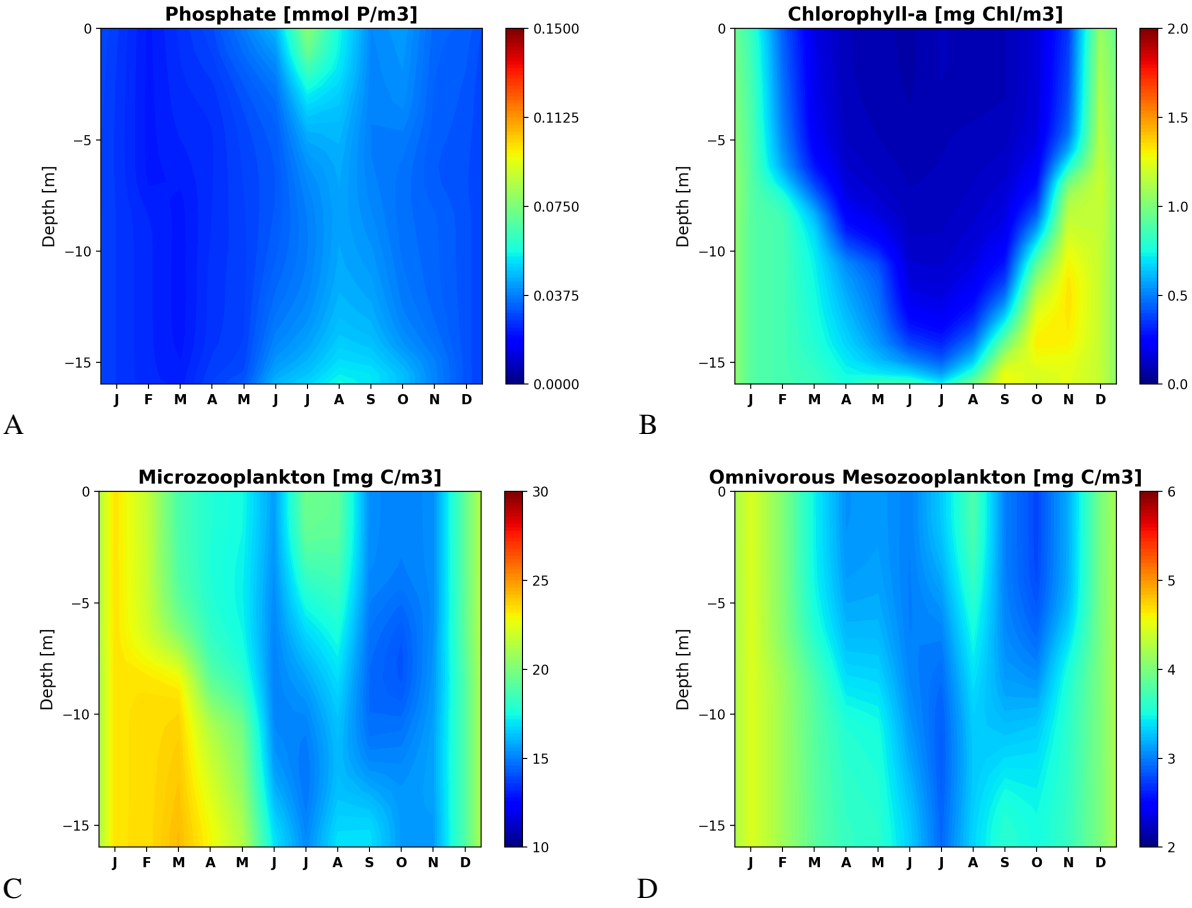


Figure 5.1: BFM-POM1D implementation in the Gulf of Trieste. Climatological annual cycles computed over the last 5 years of simulation for A) Phosphate ($mmolP/m^3$), B) Chlorophyll ($mg/m^3$), C) Microzooplankton ($mgC/m^3$), D) Omnivorous zooplankton ($mgC/m^3$).

# Bibliography


Asselin, R., 1972. Frequency filters for time integrations. Monthly Weather Review 100, 487–490.

Bianchi, D., Zavatarelli, M. andPinardi, N., Capozzi, R., Capotondi, L., C., C., S., M., 2006. Simulations of ecosystem response during the sapropel s1 deposition event. Palaeogeography, Palaeoclimatology, Palaeoecology 235, 265–287.

Blumberg, A., Mellor, G., 1987. A description of a three dimensional coastal ocean circulation model. AGU, pp. 1–16.

Butenschön, M., Zavatarelli, M., Vichi, M., 2012. Sensitivity of a marine coupled physical biogeochemical model to time resolution, integration scheme and time splitting method. Ocean Modelling 52–53, 36—53.

Galperin, B., Kantha, L., Hassid, S., Rosati, A., 1988. A quasi-equilibrium turbulent energy model for geophysical flows. Journal of Atmospheric Sciences 45, 55–62.

Hundsdorfer, W., Verwer, J., 2003. Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations. Springer Berlin / Heidelberg.

Jerlov, N., 1976. Marine Optics. Elsevier Science.

Mellor, G., 1989. Retrospect on oceanic boundary layer modeling and second moment closure. In: Hawaiian Winter Workshop on "Parameterization of Small-Scale Processes". University of Hawaii, pp. 251–271.

Mellor, G., 1991. An equation of state for numerical models of oceans and estuaries. Journal of Atmospheric and Oceanic Technology. 8, 609–611.

Mellor, G., 2004. A user guide for a three dimensional primitive equation numerical ocean model. Aos program report 34 pp., Princeton University.

Mellor, G., Yamada, T., 1982. Development of a turbulence closure model for geophysical fluid problems. Review of Geophysics and Space Physics. 20, 851–875.

Mussap, G., Zavatarelli, M., 2017. A numerical study of the benthic pelagic coupling in a shallow shelf sea (gulf of trieste). Regional Studies in Marine Science 9, 24–34.

Mussap, G., Zavatarelli, M., Pinardi, N., 2017. Linking ocean modeling to environmental management an ensemble approach. Ocean Dynamics submitted.

Mussap, G., Zavatarelli, M., Pinardi, N., Celio, M., 2016. A management oriented 1-d ecosystem model: Implementation in the gulf of trieste (adriatic sea). Regional Studies in Marine Science,, 109–103.

Paulson, C., Simpson, J., 1977. Irradiance measurements in the upper ocean. Journal of Physical Oceanography 7 (952-956).

Redfield, A., Ketchum, B., Richards, F., 1963. The influence of organisms on the composition of sea water. Vol. 2 of The Sea. Interscience N.Y., pp. 26–77.

Richards, F., 1958. Dissolved silicate and related procerties of some western north atlantic and caribbean waters. Journal of Marine Research 17, 449–465.

Richtmeyer, R., Morton, K., 1967. Difference Methods for Initial-Value Problems. Interscience N.Y.

UNESCO, 1981. The practical salinity scale 1978 and the international equation of state of seawater 1980. UNESCO technical paper in marine sciences 36, UNESCO.

Vichi, M., Lovato, T., Butenschön, M., Tedesco, L., Lazzari, P., Cossarini, G., Masina, S., Pinardi, N., Solidoro, C., Zavatarelli, M., October 2022. The Biogeochemical Flux Model (BFM): Equation Description and User Manual. BFM version 5.3. BFM Report Series 1, Bologna, Italy.
URL `http://bfm-community.eu`

Wanninkkof, R., 1992. Relationship between windspeed and gas exchange over the ocean. Journal of Geophysical Research 97, 7373–7382.