Mars_Agrif2_V9.06

“MARS manual”

Valérie GARNIER,
Bénédicte THOUVENIN,
Martin HURET

Summary

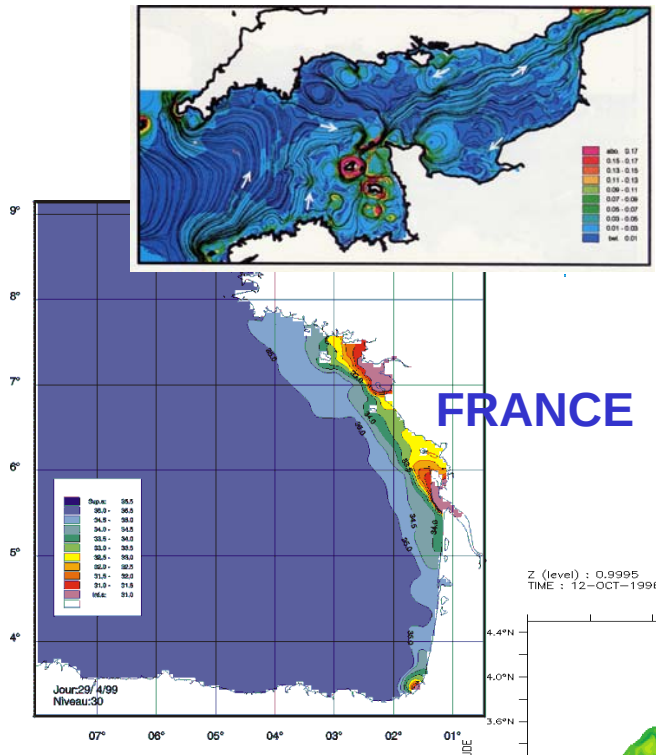
General description of MARS

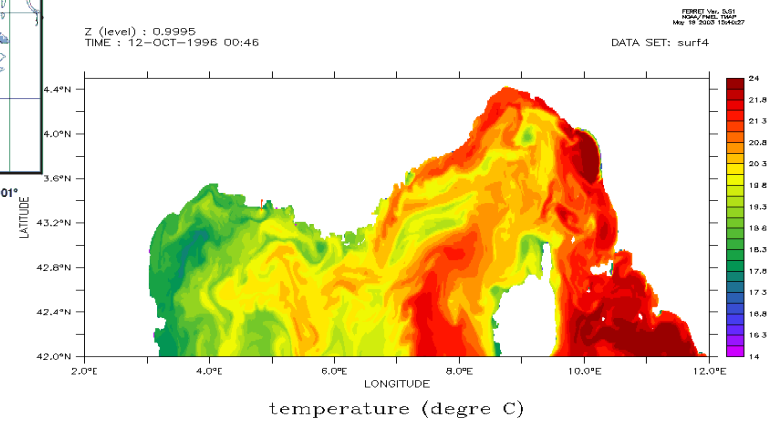
Context (1/5)

- MARS is a coastal hydrodynamical model developped by IFREMER (French Research Institute for the Exploitation of the Sea)
- In such a multidisciplinary institute, numerical researches are managed on different environmental domains.
    - Sedimentology,
    - Coastal Environment health (microbial and contaminants),
    - Biogeochemistry : primary production,…
    - Halieutic applications.
- MARS is also implemented for operational purpose : the PREVIMER project provides the users with analyses and forecasts of marine coastal parameters
- The variety of applications involves the taking into account of a large range of time-space scales :
    - basin, region, shelf, bay, beach
    - decade, season, week, tide.

Strategy of modelling at IFREMER (2/5)

MARS model is implemented for research and operational interests over the three metropolitan basins : the Channel, the Bay of Biscay and the North-Western Mediterranean Sea.



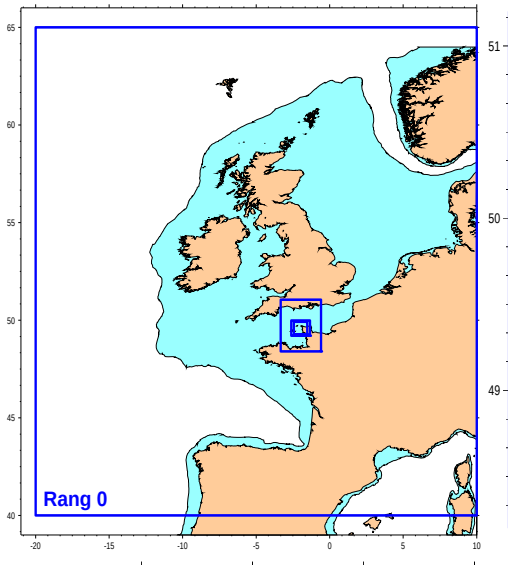


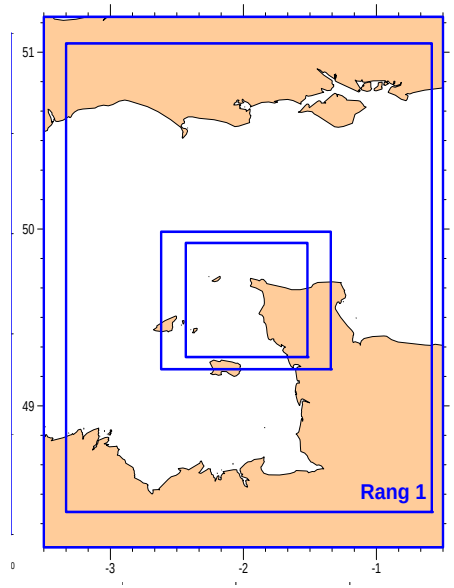
Strategy of modelling at IFREMER (3/5)

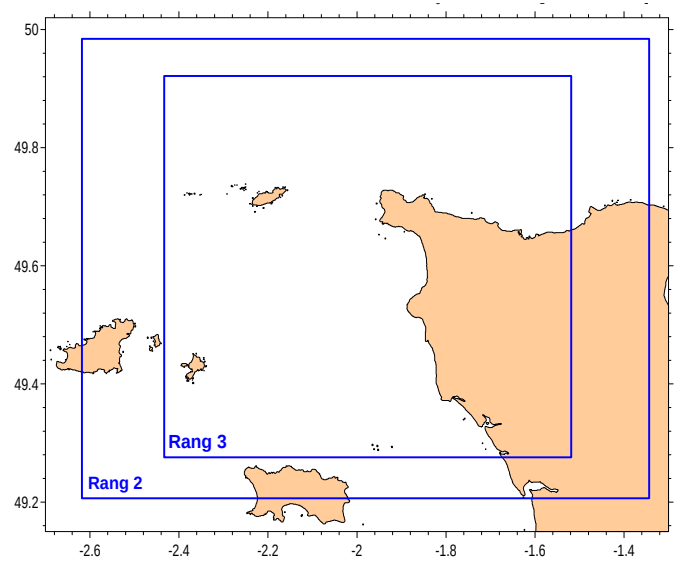
Large scales are treated by operational institutes (MFS, MERCATOR), that provides MARS with initial fields and open boundary conditions.

For the coarsest configuration (rank0), open boundary conditions for tides come from different data sets : FES2004, FES1999, Schwiderski

If required, models are coupled offline or online (AGRIF package). A configuration can be splitted into several ranks, the spatial resolution increasing with ranks

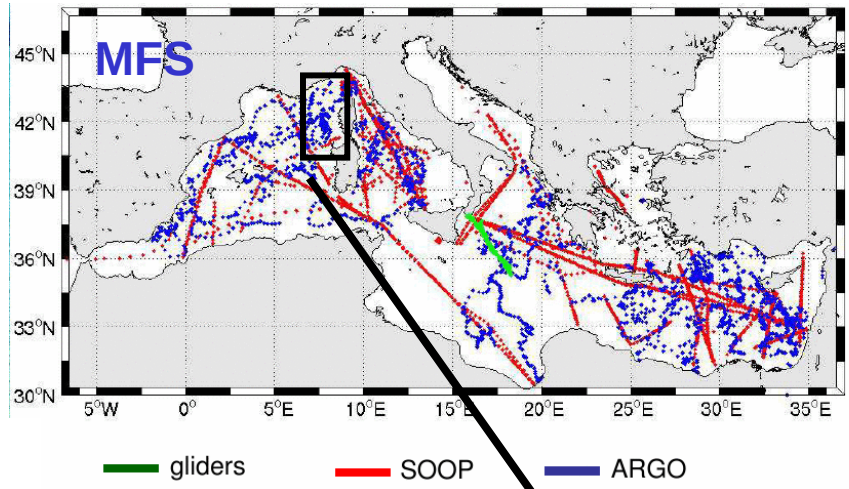


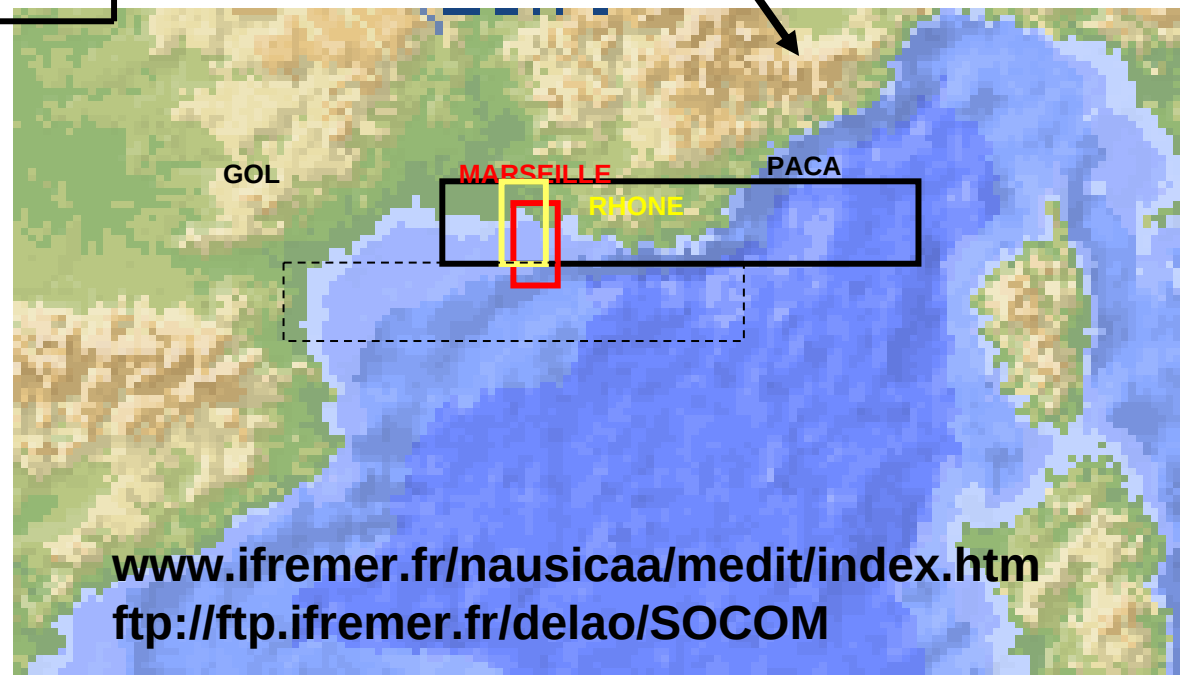




Strategy of modelling at IFREMER (4/5) Mediterranean platform

Regional configuration	Regional circulation
Coastal configurations	Exchanges with open sea Physical and sedimentological processes
Lagoons	Dynamics / survey
Local configurations	Impact studies, industrial outflow





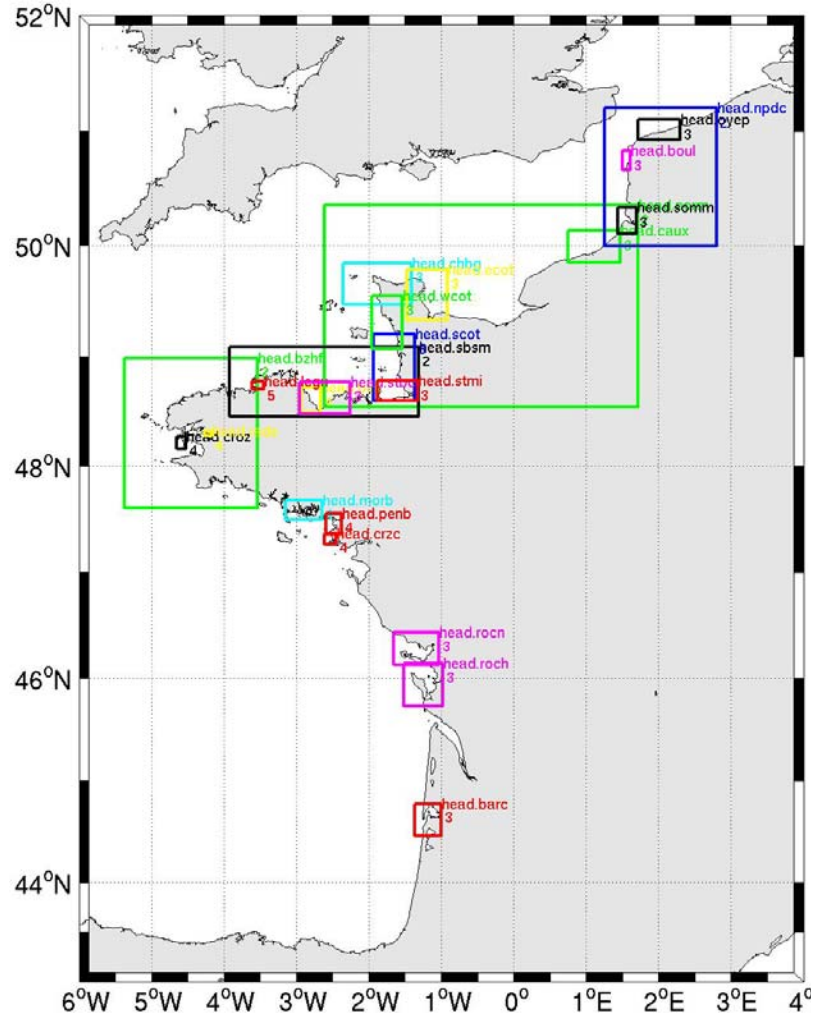
MFS (OPA : 1/16° 72z)

- Global model
- Realistic (assimilation)
- Operational

www.ifremer.fr/nausicaa/medit/index.htm
ftp://ftp.ifremer.fr/delao/SOCOM

Strategy of modelling at IFREMER (5/5)
Modelling along the Atlantic and Channel coasts

Shelf of the Bay of Biscay	Understanding of the regional circulation Hydrological structures Seasonal and interannual changes
Channel	Validation of residual fields Tracking of water masses Outflow (La Hague) Biological applications
Local configuration	Impact studies Impact studies, industrial and urban outflows Microbian state in microtidal seas



More than 30 configurations are implemented by Coastal Environment Laboratories

Equations

Primitive equations (z coordinate) (1/7)

Navier-Stockes equations

+ Boussinesq approximation

- **ρ' << ρ0**

  incompressible equation of continuity,

  ρ ~ ρ0 in momentum equation except for the buoyancy term

+ hydrostatic assumption

- **H << L**

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}-fv=-g\frac{\partial \zeta}{\partial x}-\frac{1}{\rho_0}\frac{\partial Pa}{\partial x}+\frac{1}{\rho_0}\frac{\partial p'}{\partial x}+\frac{\partial (nz\frac{\partial u}{\partial z})}{\partial z}+F_x$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}+fu=-g\frac{\partial \zeta}{\partial y}-\frac{1}{\rho_0}\frac{\partial Pa}{\partial y}+\frac{1}{\rho_0}\frac{\partial p'}{\partial y}+\frac{\partial (nz\frac{\partial v}{\partial z})}{\partial z}+F_y$$

(tangent plane)

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0$$

$$\frac{\partial p'}{\partial z}=\rho_0 b \quad \textbf{with} \quad b=-g(\rho-\rho_0)/\rho_0$$

+ equations of thermodynamics

$$\frac{\partial T}{\partial t}+\frac{\partial (uT-k_x\frac{\partial T}{\partial x})}{\partial x}+\frac{\partial (vT-k_y\frac{\partial T}{\partial y})}{\partial y}+\frac{\partial (wT-k_z\frac{\partial T}{\partial z})}{\partial z}=\frac{1}{\rho_0 C_p}\frac{\partial I}{\partial z}$$

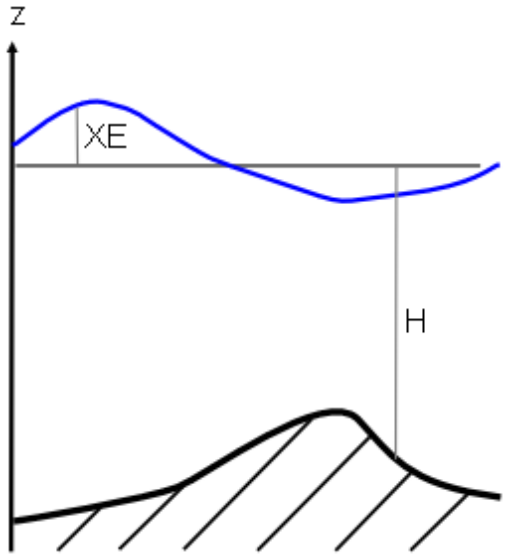
$$\frac{\partial S}{\partial t}+\frac{\partial (uS-k_x\frac{\partial S}{\partial x})}{\partial x}+\frac{\partial (vS-k_y\frac{\partial S}{\partial y})}{\partial y}+\frac{\partial (wS-k_z\frac{\partial S}{\partial z})}{\partial z}=0$$

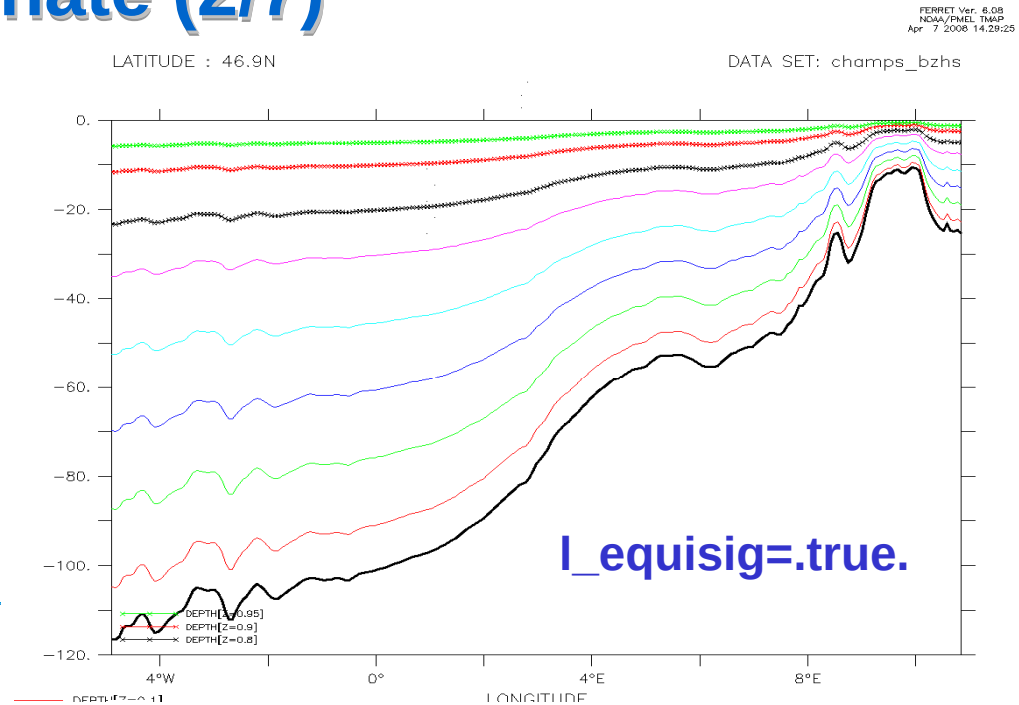
σ coordinate (2/7)

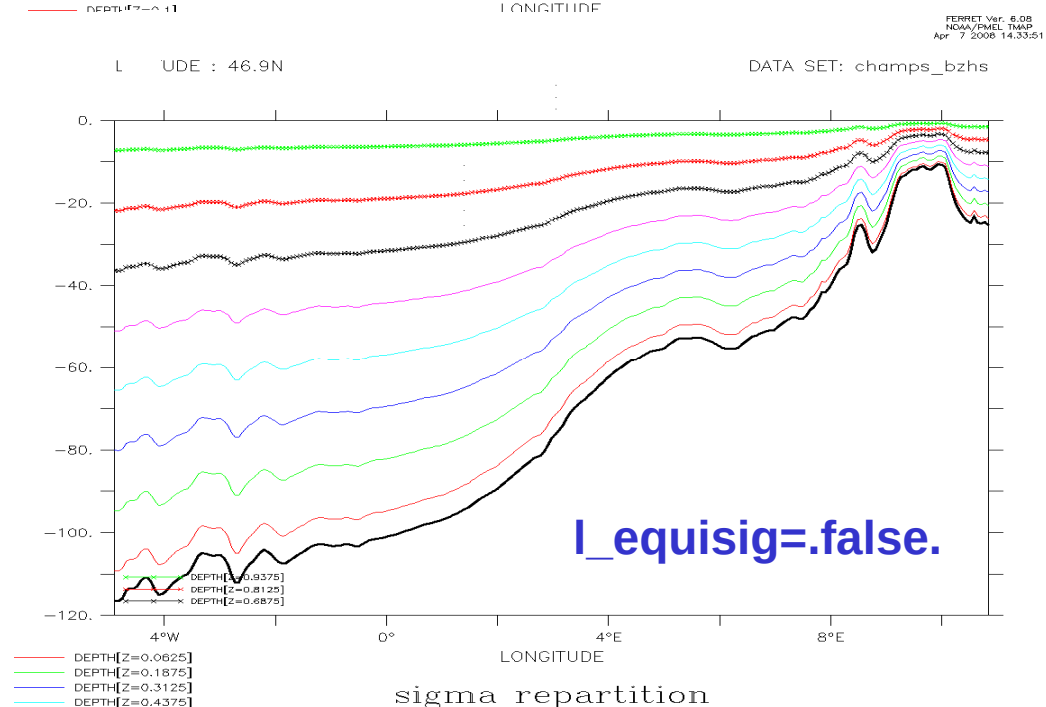
$$\sigma = \frac{z - xe}{h0 + xe}$$

σ = 0 at the sea surface

σ = -1 at the sea floor







Generalized σ coordinate (3/7)

l_equisig=.true.
key_siggen

σ = 0

σ = -1

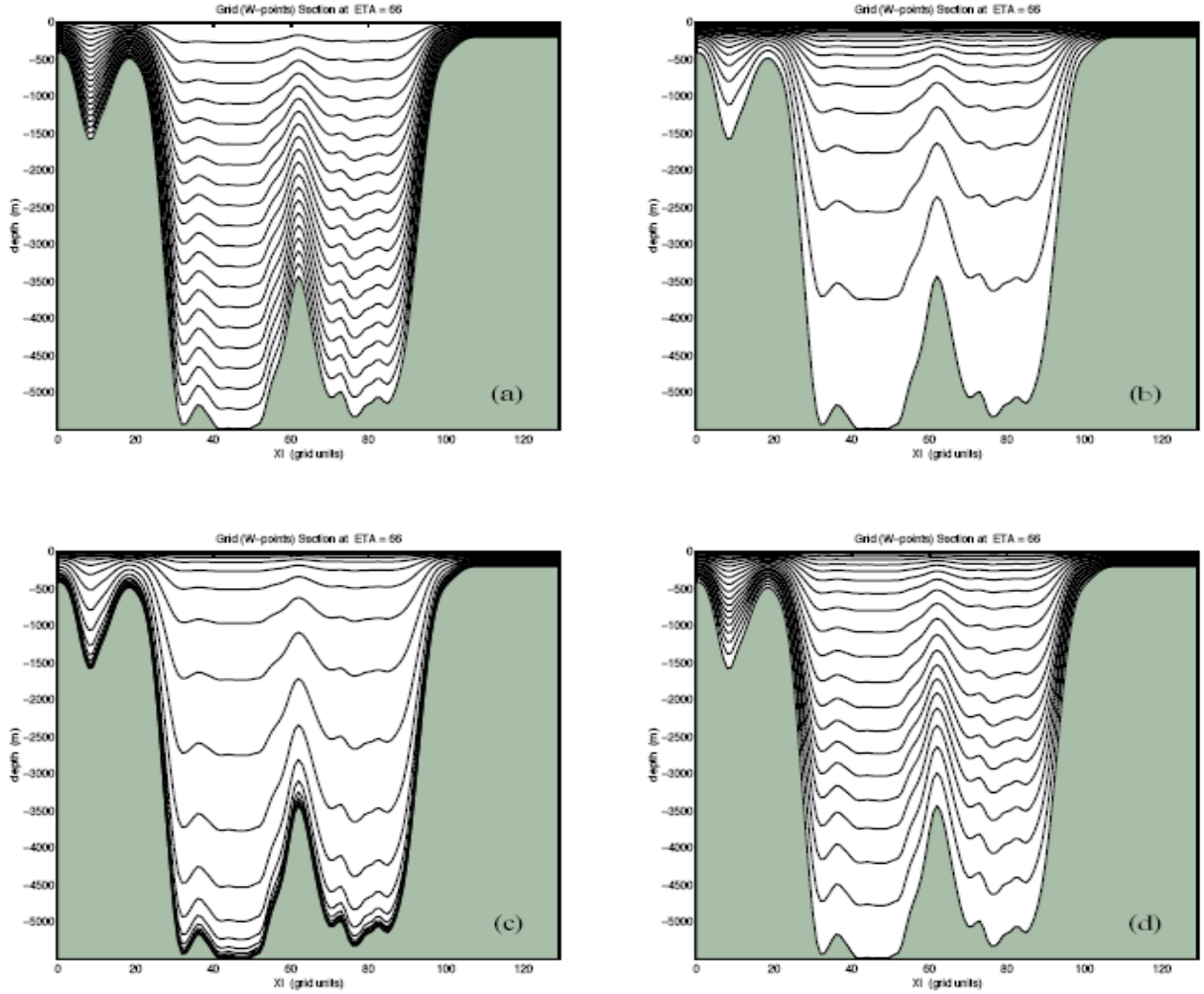


Figure 23: The s-surfaces for the North Atlantic with (a) $\theta = 0.0001$ and $b = 0$, (b) $\theta = 8$ and $b = 0$, (c) $\theta = 8$ and $b = 1$. (d) The actual values used in this domain were $\theta = 5$ and $b = 0.4$.

Generalized σ coordinate (4/7)

Following Song and Haidvogel [54], the vertical coordinate has been chosen to be:

$$z = \zeta(1+s) + h_c s + (h - h_c)C(s), \qquad -1 \le s \le 0 \tag{221}$$

where h_c is either the minimum depth or a shallower depth above which we wish to have more resolution. $C(s)$ is defined as:

$$C(s) = (1-b)\frac{\sinh(\theta s)}{\sinh\theta} + b\frac{\tanh[\theta(s+\frac{1}{2})] - \tanh(\frac{1}{2}\theta)}{2\tanh(\frac{1}{2}\theta)} \tag{222}$$

where θ and b are surface and bottom control parameters. Their ranges are $0 < \theta \le 20$ and $0 \le b \le 1$, respectively. Equation (221) leads to $z = \zeta$ for $s = 0$ and $z = h$ for $s = -1$.

Some features of this coordinate system:

- It is a generalization of the σ-coordinate system. Letting θ go to zero and using L'Hopital's rule, we get:

$$z = (\zeta + h)(1+s) - h \tag{223}$$

  which is the σ-coordinate.

- It has a linear dependence on ζ and is infinitely differentiable in s.
- The larger the value of θ, the more resolution is kept above h_c.
- For $b = 0$, the resolution all goes to the surface as θ is increased.
- For $b = 1$, the resolution goes to both the surface and the bottom equally as θ is increased.
- For $\theta \neq 0$ there is a subtle mismatch in the discretization of the model equations, for instance in the horizontal viscosity term. We recommend that you stick with "reasonable" values of θ, say $\theta \le 5$.
- Some problems turn out to be sensitive to the value of θ used.

Primitive equations (σ coordinate) (5/7)

$$\frac{\partial u}{\partial t}+L(u)-fv=-g\frac{\partial \zeta}{\partial x}-\frac{1}{\rho_0}\frac{\partial Pa}{\partial x}+\pi_x+\frac{1}{D}\frac{\partial(\frac{nz}{D}\frac{\partial u}{\partial \sigma})}{\partial \sigma}+F_x$$

$$\frac{\partial v}{\partial t}+L(v)+fu=-g\frac{\partial \zeta}{\partial y}-\frac{1}{\rho_0}\frac{\partial Pa}{\partial y}+\pi_y+\frac{1}{D}\frac{\partial(\frac{nz}{D}\frac{\partial v}{\partial \sigma})}{\partial \sigma}+F_y$$

with $$L(A)=u\frac{\partial A}{\partial x}+v\frac{\partial A}{\partial y}+\overset{\text{å}}{w}\frac{\partial A}{\partial \sigma}$$

$$\frac{1}{D}\frac{\partial p}{\partial \sigma}=-\rho g$$

$$\frac{\partial \zeta}{\partial t}+\frac{\partial Du}{\partial x}+\frac{\partial Dv}{\partial y}+\frac{\partial D\overset{\text{å}}{w}}{\partial \sigma}=0$$

with $$\overset{\text{å}}{w}=\frac{1}{D}(w-\sigma\frac{\partial \zeta}{\partial t}-u(\sigma\frac{\partial \zeta}{\partial x}+(\sigma-1)\frac{\partial H}{\partial x})-v(\sigma\frac{\partial \zeta}{\partial y}+(\sigma-1)\frac{\partial H}{\partial y}))$$

To simplify, equations are written using cartesian coordinates in this documentation, but MARS also uses spherical coordinates

Primitive equations (σ coordinate) (6/7)

$$f = 2\Omega sin\phi$$

Coriolis parameter

$$L(A) = u\frac{\partial A}{\partial x} + v\frac{\partial A}{\partial y} + w^{å}\frac{\partial A}{\partial \sigma}$$

advection

$$\pi_x = \frac{\partial}{\partial x}[D\int_{\sigma}^{1} b\, d\sigma] + b(\sigma\frac{\partial D}{\partial x} - \frac{\partial H}{\partial x})$$

$$\pi_y = \frac{\partial}{\partial y}[D\int_{\sigma}^{1} b\, d\sigma] + b(\sigma\frac{\partial D}{\partial y} - \frac{\partial H}{\partial y})$$

internal pressure

$$b = -g(\rho - \rho_0)/\rho_0$$

buoyancy

Horizontal friction

$$F_x = \frac{1}{D}\frac{\partial}{\partial x}[D\nu_x\frac{\partial u}{\partial x}] + \frac{1}{D}\frac{\partial}{\partial x}[\nu_x(\frac{\partial H}{\partial x} - \sigma\frac{\partial D}{\partial x})\frac{\partial u}{\partial \sigma}] +$$

$$\frac{1}{D}\frac{\partial}{\partial \sigma}[\nu_x(\frac{\partial H}{\partial x} - \sigma\frac{\partial D}{\partial x})\frac{\partial u}{\partial x}] + \frac{1}{D}\frac{\partial}{\partial \sigma}[\frac{\nu_x}{D}(\frac{\partial H}{\partial x} - \sigma\frac{\partial D}{\partial x})^2\frac{\partial u}{\partial \sigma}]$$

$$F_y = \frac{1}{D}\frac{\partial}{\partial y}[D\nu_y\frac{\partial v}{\partial y}] + \frac{1}{D}\frac{\partial}{\partial y}[\nu_y(\frac{\partial H}{\partial y} - \sigma\frac{\partial D}{\partial y})\frac{\partial v}{\partial \sigma}] +$$

$$\frac{1}{D}\frac{\partial}{\partial \sigma}[\nu_y(\frac{\partial H}{\partial y} - \sigma\frac{\partial D}{\partial y})\frac{\partial v}{\partial y}] + \frac{1}{D}\frac{\partial}{\partial \sigma}[\frac{\nu_y}{D}(\frac{\partial H}{\partial y} - \sigma\frac{\partial D}{\partial y})^2\frac{\partial v}{\partial \sigma}]$$

Smooth bathymetric gradient

$$F_x = \frac{\partial}{\partial x}(\nu_x\frac{\partial u}{\partial x}) \quad F_y = \frac{\partial}{\partial y}(\nu_y\frac{\partial v}{\partial y})$$

Equations of thermodynamics (7/7)

Heat and salinity equations

$$\frac{\partial DT}{\partial t}+\frac{\partial D(uT-k_x\frac{\partial T}{\partial x})}{\partial x}+\frac{\partial D(vT-k_y\frac{\partial T}{\partial y})}{\partial y}+\frac{\partial D(w^{\mathring{a}}T-\frac{kz}{D^2}\frac{\partial T}{\partial \sigma})}{\partial \sigma}=\frac{1}{\rho_0 C_p}\frac{\partial I}{\partial \sigma}$$

$$\frac{\partial DS}{\partial t}+\frac{\partial D(uS-k_x\frac{\partial S}{\partial x})}{\partial x}+\frac{\partial D(vS-k_y\frac{\partial S}{\partial y})}{\partial y}+\frac{\partial D(w^{\mathring{a}}S-\frac{kz}{D^2}\frac{\partial S}{\partial \sigma})}{\partial \sigma}=0$$

Equation of state

$$\rho=F(S,T,p)$$

- Linearized around point T_0,S_0,p=0
- Fully developped as Mellor (1991)

Passive tracer

$$\frac{\partial DT_r}{\partial t}+\frac{\partial D(uT_r-k_x\frac{\partial T_r}{\partial x})}{\partial x}+\frac{\partial D(vT_r-k_y\frac{\partial T_r}{\partial y})}{\partial y}+\frac{\partial D(w^{\mathring{a}}T_r-\frac{kz}{D^2}\frac{\partial T_r}{\partial \sigma})}{\partial \sigma}=Sources-Sinks$$

Forcings

Momentum equations
- •Winds, tides
- •River discharges (mass)

Thermodynamics
- •Heat fluxes (LW, SH, latent, sensible)
- •River discharges (salinity)

Passive tracer
- •Sewage…

$$\frac{\partial \zeta}{\partial t} + \frac{\partial D\bar{u}}{\partial x} + \frac{\partial D\bar{v}}{\partial y} = \frac{Q}{surf}$$

$$w = -Q/(surf * D)$$

Boundary conditions at the surface $\sigma = 1$	Boundary conditions at the bottom $\sigma = 0$
$\frac{n_z}{D}\frac{\partial u}{\partial \sigma} = \frac{\tau_{sx}}{\rho_0}$	$\frac{n_z}{D}\frac{\partial u}{\partial \sigma} = \frac{\tau_{bx}}{\rho_0}$
$\frac{n_z}{D}\frac{\partial v}{\partial \sigma} = \frac{\tau_{sy}}{\rho_0}$	$\frac{n_z}{D}\frac{\partial v}{\partial \sigma} = \frac{\tau_{by}}{\rho_0}$
$\frac{k_z}{D}\frac{\partial T}{\partial \sigma} = \frac{Q_T}{\rho_0 C_p}$	$k_z \frac{\partial T}{\partial \sigma} = 0$
$k_z \frac{\partial S}{\partial \sigma} = 0$	$k_z \frac{\partial S}{\partial \sigma} = 0$
$w^{å} = 0$	$w^{å} = 0$

$$(\tau_{sx}, \tau_{sy}) = \rho_a Cd_S \left\|\vec{W}\right\| (W_x, W_y)$$

$$(\tau_{bx}, \tau_{by}) = \rho_0 Cd_B \left\|\vec{u}\right\| (u, v) \quad Cd_B = \left(\frac{\kappa}{\ln\left(\frac{z + h + z_0}{z_0}\right)} \right)^2$$

Vertical mixing (1/2)

$$\frac{1}{D}\frac{\partial(\frac{nz}{D}\frac{\partial u}{\partial \sigma})}{\partial \sigma} \qquad nz, kz$$

Turbulence models

0 equations :

cst

Prandt model $nz = u^* H / 15\sigma\sqrt{(1-\sigma)} \quad ; \quad u^* = 0.4 / \log(\sigma H / z_0) u_{bottom}$

Quetin model $nz = l^2 \frac{\partial u}{\partial z} \quad ; \quad kz = f(Ri) l^2 \frac{\partial u}{\partial z} \quad ; l \, analytical$

Pacanovski et Philander, 1981 $nz = \frac{\nu_0}{(1+\alpha Ri)^n} + \nu_b \quad ; \quad kz = \frac{\nu}{(1+\alpha Ri)} + k_b$

$$Ri = \frac{\partial b}{\partial z} / \left|\frac{\partial U}{\partial z}\right|^2$$

1 equation : Gaspard et al, 1990 $nz, kz = f(ect, l) \quad ; \quad ect \, computed \; ; \quad l \, evaluated$

2 equations : $nz, kz = f(k, kl, l) \quad ; \quad k \, et \, kl \, computed \; ; \quad l \, evaluated$

Vertical mixing (2/2)

$$\frac{1}{D}\frac{\partial(\frac{nz}{D}\frac{\partial u}{\partial \sigma})}{\partial \sigma} \qquad nz, kz$$

Turbulence models

2 equations : Warner et al. (2005)

4 **turbulence closure models**

1. **k-kl**
   - Close to Mellor–Yamada level 2.5 scheme of Mellor et Yamada (1974, 1982)
2. **k–ε**
   - Jones and Launder (1972), Launder and Sharma (1974)
   - Burchard et al. (1998), Burchard and Bolding (2001)
3. **k-ω**
   - Kolmogorov (1942), Saffman (1970), Saffman and Wilcox (1974), Umlauf et al. (2003)
4. **GLS : generic length scale**
   - Umlauf and Burchard (2003)

Numerical formulation

Mode splitting

Motivation

Pressure gradient can be splitted in 3 terms : atmospherical pressure, sea surface level, internal pressure gradient (buoyancy). The two first ones are barotropic terms (external mode) associated with fast propagating waves (tide waves, storm surges…).

Explicit formulation (OPA, HYCOM, POM)

External and internal modes are splitted and resolved separately.
The simulation of external mode requires the use of small time steps in order to satisfy the CFL criteria.
The internal mode is resolved with a longer time step.
Splitting mode is mathematically correct for linear equations only. Because fully equations are not linear, a coupling method is required to fit barotropic and baroclinic modes.

Semi-implicit formulation (MARS)

A unique time step, allowing a more precise coupling procedure.

Time stepping : external mode (1/6)

Barotropic mode is solved using a time stepping and Alternate Direction Implicit (ADI) methodology (Peaceman & Rachford, 1995, Leendertse,1970).

Advantage : the external gravity wave term, semi-implicit, removes the stringest stability criterion

$$\Delta t < \Delta x / \sqrt{gD}$$

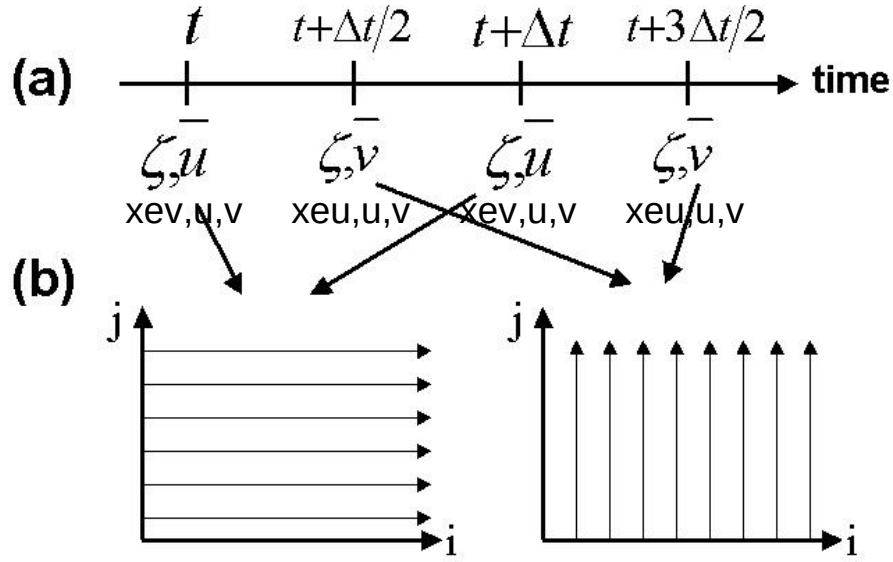
that becomes

$$C = \frac{u_{\max} \Delta t}{\Delta x} \leq C_{crit} \quad avec \quad C_{crit} = 0.7$$

and allows larger time steps

Drawback : increasing the time step too far beyond the explicit limit results in damping of the barotropic mode

In MARS, scheme are semi-implicit because implicit only with respect to the direction of computation.



Lazure & Dumas, 2008

Time stepping in 2D code in dyn2dxy.F90 (2/6)

$$
\begin{cases}
(1+ft\beta_2\Delta t)u^{n+\frac{1}{2},*} = u^n - \Delta t(u^n\frac{\partial u^n}{\partial x}+v^n\frac{\partial u^n}{\partial y}) - g\Delta t\frac{\partial \zeta^n}{\partial x} + \Delta t f v^n \\
\quad +\Delta t\left\{\frac{\partial}{\partial x}\left[\nu\frac{\partial u^n}{\partial x}\right]+\frac{\partial}{\partial y}\left[\nu\frac{\partial u^n}{\partial y}\right]\right\} \\
\quad +\Delta t\frac{\tau_{sx}}{\rho h^n} - \frac{\Delta t}{\rho}\frac{\partial Pa}{\partial x} - \Delta t f t\beta_1 u^n \\
(1+ft\beta_2\Delta t)v^{n+\frac{1}{2},*} = v^n - \Delta t(u^n\frac{\partial v^n}{\partial x}+v^n\frac{\partial v^n}{\partial y}) - g\Delta t\frac{\partial \zeta^n}{\partial y} - \Delta t f u^n \\
\quad +\Delta t\left\{\frac{\partial}{\partial x}\left[\nu\frac{\partial v^n}{\partial x}\right]+\frac{\partial}{\partial y}\left[\nu\frac{\partial v^n}{\partial y}\right]\right\} \\
\quad +\Delta t\frac{\tau_{sy}}{\rho h^n} - \frac{\Delta t}{\rho}\frac{\partial Pa}{\partial y} - \Delta t f t\beta_1 v^n
\end{cases}
$$

Explicit estimates of velocities
(the bottom stress is the only one semi-implicit term)
dyn2d_uvstar

$$
\begin{cases}
\zeta^{n+\frac{1}{2},*} = \zeta^n - \Delta t\frac{\partial}{\partial x}\left[h^n\left(\alpha\, u^n + (1-\alpha)\, u^{n+\frac{1}{2}}\right)\right] \\
\quad - \Delta t\frac{\partial}{\partial y}\left[h^n\left(\alpha\, v^n + (1-\alpha)\, v^{n+\frac{1}{2},*}\right)\right] \\
(1+ft\beta_2\Delta t)u^{n+\frac{1}{2}} = u^n - g\Delta t\frac{\partial}{\partial x}\left(\alpha\,\zeta^n + (1-\alpha)\,\zeta^{n+\frac{1}{2},*}\right) + \Delta t f v^{n+\frac{1}{2},*} - \Delta t f t\beta_1 u^n \\
\quad - \Delta t u^{n+\frac{1}{2},*}\frac{\partial}{\partial x}\left(\alpha\, u^n + (1-\alpha)\, u^{n+\frac{1}{2},*}\right) \\
\quad - \Delta t v^{n+\frac{1}{2},*}\frac{\partial}{\partial y}\left(\alpha\, u^n + (1-\alpha)\, u^{n+\frac{1}{2},*}\right) \\
\quad + \Delta t\left\{\frac{\partial}{\partial x}\left[\nu\frac{\partial u^n}{\partial x}\right]+\frac{\partial}{\partial y}\left[\nu\frac{\partial u^n}{\partial y}\right]\right\} \\
\quad + \Delta t\frac{\tau_{sx}}{h^n\rho} - \frac{\Delta t}{\rho}\frac{\partial Pa}{\partial x}
\end{cases}
$$

barotropic tridiagonal linear system along x
Coefficients and RHS : dyn2dx_tls_coef
(dyn2dx_cty, dyn2dx_mtm)
Resolving : dyn2dx_tls_solve

$$
\begin{cases}
\zeta^{n+\frac{1}{2}} = \zeta^n - \Delta t\frac{\partial}{\partial x}\left[\left(\alpha h^n + (1-\alpha)h^{n+\frac{1}{2},*}\right)\left(\alpha\, u^n + (1-\alpha)\, u^{n+\frac{1}{2}}\right)\right] \\
\quad - \Delta t\frac{\partial}{\partial y}\left[\left(\alpha h^n + (1-\alpha)h^{n+\frac{1}{2},*}\right)\left(\alpha\, v^n + (1-\alpha)\, v^{n+\frac{1}{2}}\right)\right] \\
(1+ft\beta_2\Delta t)v^{n+\frac{1}{2}} = v^n - g\Delta t\frac{\partial}{\partial y}\left(\alpha\,\zeta^n + (1-\alpha)\,\zeta^{n+\frac{1}{2}}\right) - f\Delta t u^{n+\frac{1}{2}} - \Delta t f t\beta_1 v^n \\
\quad - \Delta t u^{n+\frac{1}{2},*}\frac{\partial}{\partial x}\left(\alpha\, v^n + (1-\alpha)\, v^{n+\frac{1}{2},*}\right) \\
\quad - \Delta t v^{n+\frac{1}{2},*}\frac{\partial}{\partial y}\left(\alpha\, v^n + (1-\alpha)\, v^{n+\frac{1}{2},*}\right) \\
\quad + \Delta t\left\{\frac{\partial}{\partial x}\left[\nu\frac{\partial v^n}{\partial x}\right]+\frac{\partial}{\partial y}\left[\nu\frac{\partial v^n}{\partial y}\right]\right\} \\
\quad + \Delta t\frac{\tau_{sy}}{\rho\left(\alpha\, h^n + (1-\alpha)\, h^{n+\frac{1}{2},*}\right)} - \frac{\Delta t}{\rho}\frac{\partial Pa}{\partial y}
\end{cases}
$$

barotropic tridiagonal linear system along y
Coefficients and RHS : dyn2dy_ctymtm2
Resolving : dyn2dy_tls_solve

Balance between sea surface height and meridional velocity estimates at n+1/2

Time stepping in 2D code in dyn2dyx.F90 (3/6)

$$
\begin{cases}
(1+ft\beta_2\Delta t)u^{n+1,*} = u^{n+\frac{1}{2}} - \Delta t(u^{n+\frac{1}{2}}\frac{\partial u^{n+\frac{1}{2}}}{\partial x} + v^{n+\frac{1}{2}}\frac{\partial u^{n+\frac{1}{2}}}{\partial y}) - g\Delta t\frac{\partial \zeta^{n+\frac{1}{2}}}{\partial x} + \Delta t f v^{n+\frac{1}{2}} \\
\quad +\Delta t\left\{\frac{\partial}{\partial x}\left[\nu\frac{\partial u^{n+\frac{1}{2}}}{\partial x}\right] + \frac{\partial}{\partial y}\left[\nu\frac{\partial u^{n+\frac{1}{2}}}{\partial y}\right]\right\} \\
\quad +\Delta t\frac{\tau_{sx}}{\rho h} - \Delta t\frac{1}{\rho}\frac{\partial Pa}{\partial x} - \Delta t f t\beta_1 u^{n+\frac{1}{2}} \\
(1+ft\beta_2\Delta t)v^{n+1,*} = v^{n+\frac{1}{2}} - \Delta t(u^{n+\frac{1}{2}}\frac{\partial v^{n+\frac{1}{2}}}{\partial x} + v^{n+\frac{1}{2}}\frac{\partial v^{n+\frac{1}{2}}}{\partial y}) - g\Delta t\frac{\partial \zeta^{n+\frac{1}{2}}}{\partial y} - \Delta t f u^{n+\frac{1}{2}} \\
\quad +\Delta t\left\{\frac{\partial}{\partial x}\left[\nu\frac{\partial v^{n+\frac{1}{2}}}{\partial x}\right] + \frac{\partial}{\partial y}\left[\nu\frac{\partial v^{n+\frac{1}{2}}}{\partial y}\right]\right\} \\
\quad +\Delta t\frac{\tau_{sy}}{\rho h} - \frac{\Delta t}{\rho}\frac{\partial Pa}{\partial y} - \Delta t f t\beta_1 v^{n+\frac{1}{2}}
\end{cases}
$$

Explicit estimates of velocities
(the bottom stress is the only one semi-implicit term)
dyn2d_uvstar

$$
\begin{cases}
\zeta^{n+1,*} = \zeta^{n+\frac{1}{2}} - \Delta t\frac{\partial}{\partial x}\left[h^{n+\frac{1}{2}}\left(\alpha\, u^{n+\frac{1}{2}} + (1-\alpha)\, u^{n+1,*}\right)\right] \\
\quad - \Delta t\frac{\partial}{\partial y}\left[h^{n+\frac{1}{2}}\left(\alpha\, v^{n+\frac{1}{2}} + (1-\alpha)\, v^{n+1}\right)\right] \\
(1+ft\beta_2\Delta t)v^{n+1} = v^{n+\frac{1}{2}} - g\Delta t\frac{\partial}{\partial y}\left(\alpha\,\zeta^{n+\frac{1}{2}} + (1-\alpha)\,\zeta^{n+1,*}\right) - f\Delta t u^{n+1,*} - \Delta t f t\beta_1 v^{n+\frac{1}{2}} \\
\quad - \Delta t u^{n+1,*}\frac{\partial}{\partial x}\left(\alpha\, v^{n+\frac{1}{2}} + (1-\alpha)\, v^{n+1,*}\right) \\
\quad - \Delta t v^{n+1,*}\frac{\partial}{\partial y}\left(\alpha\, v^{n+\frac{1}{2}} + (1-\alpha)\, v^{n+1,*}\right) \\
\quad + \Delta t\left\{\frac{\partial}{\partial x}\left[\nu\frac{\partial v^{n+\frac{1}{2}}}{\partial x}\right] + \frac{\partial}{\partial y}\left[\nu\frac{\partial v^{n+\frac{1}{2}}}{\partial y}\right]\right\} \\
\quad + \Delta t\frac{\tau_{sy}}{\rho h^{n+\frac{1}{2}}} - \frac{\Delta t}{\rho}\frac{\partial Pa}{\partial y}
\end{cases}
$$

barotropic tridiagonal linear system along y
Coefficients and RHS : dyn2dy_tls_coef
dyn2dy_cty, dyn2dy_mtm
Resolving : dyn2dy_tls_solve

$$
\begin{cases}
\zeta^{n+1} = \zeta^{n+\frac{1}{2}} - \Delta t\frac{\partial}{\partial x}\left[\left(\alpha h^{n+\frac{1}{2}} + (1-\alpha)h^{n+1,*}\right)\left(\alpha\, u^{n+\frac{1}{2}} + (1-\alpha)\, u^{n+1}\right)\right] \\
\quad - \Delta t\frac{\partial}{\partial y}\left[\left(\alpha h^{n+\frac{1}{2}} + (1-\alpha)h^{n+1,*}\right)\left(\alpha\, v^{n+\frac{1}{2}} + (1-\alpha)\, v^{n+1}\right)\right] \\
(1+ft\beta_2\Delta t)u^{n+1} = u^{n+\frac{1}{2}} - g\Delta t\frac{\partial}{\partial x}\left(\alpha\,\zeta^{n+\frac{1}{2}} + (1-\alpha)\,\zeta^{n+1}\right) + \Delta t f v^{n+1} - \Delta t f t\beta_1 u^{n+\frac{1}{2}} \\
\quad - \Delta t u^{n+1,*}\frac{\partial}{\partial x}\left(\alpha\, u^{n+\frac{1}{2}} + (1-\alpha)\, u^{n+1,*}\right) \\
\quad - \Delta t v^{n+1,*}\frac{\partial}{\partial y}\left(\alpha\, u^{n+\frac{1}{2}} + (1-\alpha)\, u^{n+1,*}\right) \\
\quad + \Delta t\left\{\frac{\partial}{\partial x}\left[\nu\frac{\partial u^{n+\frac{1}{2}}}{\partial x}\right] + \frac{\partial}{\partial y}\left[\nu\frac{\partial u^{n+\frac{1}{2}}}{\partial y}\right]\right\} \\
\quad + \Delta t\frac{\tau_{sx}}{\rho\left(\alpha h^{n+\frac{1}{2}} + (1-\alpha)h^{n+1,*}\right)} - \frac{\Delta t}{\rho}\frac{\partial Pa}{\partial x}
\end{cases}
$$

barotropic tridiagonal linear system along x
Coefficients and RHS : dyn2dx_ctymtm2
Resolving : dyn2dx_tls_solve

Balance between sea surface height and zonal velocity estimates at n+1/2

Time stepping and 3D coupling procedure in dyn3dxy.F90 (4/6)

from dyn3dy

$$\left\{ \bar{v}^{n+\frac{1}{2},*} = \bar{v}^{n} - \Delta t \left[\bar{G}_v \left(u^{n-\frac{1}{2}}, v^{n-\frac{1}{2}} \right) + g \frac{\partial \zeta^n}{\partial y} \right] \right.$$

Time extrapolation, $G^{n-1/2}$ to use old estimate

dyn3dx_tls_coef
dyn3dx_cty
dyn3dx_mtm

$$\left\{ \begin{array}{rcl} \zeta^{n+\frac{1}{2},*} & = & \zeta^{n} - \Delta t \left[\frac{\partial}{\partial x} \left(h^{n} u^{n+\frac{1}{2}} \right) + \frac{\partial}{\partial y} \left(h^{n} v^{n+\frac{1}{2},*} \right) \right] \\ \bar{u}^{n+\frac{1}{2}} & = & \bar{u}^{n} - \Delta t \left[g \frac{\partial}{\partial x} \left(\alpha \zeta^{n} + (1-\alpha) \zeta^{n+\frac{1}{2},*} \right) + \bar{G}_u \left(u^n, v^n \right) \right] \\ u^{n+\frac{1}{2}}(k) & = & u^{n}(k) - \Delta t \left[g \frac{\partial}{\partial x} \left(\alpha \zeta^{n} + (1-\alpha) \zeta^{n+\frac{1}{2},*} \right) + G_u \left(u^n, v^n \right)(k) \right] \\ & & + \Delta t \frac{\partial}{\partial z} \left[k_z \frac{\partial}{\partial z} \left(u^{n+\frac{1}{2}}(k) \right) \right] \end{array} \right.$$

Solve the barotropic tridiagonal linear system dyn2dx_tls_solve

Solve the tridiagonal linear system along z dyn3dx_ztls_solve

$$\left\{ u^{n+\frac{1}{2}}(k) = u^{n+\frac{1}{2}}(k) - \overline{u^{n+\frac{1}{2}}(k)} + \bar{u}^{n+\frac{1}{2}} \right.$$

3D coupling

dyn3d_height_mask

dyn3dy_tls_coef
dyn3dy_cty
dyn3dy_mtm

$$\left\{ \begin{array}{rcl} \zeta^{n+\frac{1}{2}} & = & \zeta^{n} - \Delta t \left\{ \frac{\partial}{\partial x} \left[\left(\alpha h^{n} + (1-\alpha) h^{n+\frac{1}{2},*} \right) \bar{u}^{n+\frac{1}{2}} \right] \right. \\ & & \left. + \frac{\partial}{\partial y} \left[\left(\alpha h^{n} + (1+\alpha) h^{n+\frac{1}{2},*} \right) \bar{v}^{n+\frac{1}{2}} \right] \right\} \\ \bar{v}^{n+\frac{1}{2}} & = & \bar{v}^{n} - \Delta t \left[g \frac{\partial}{\partial y} \left(\alpha \zeta^{n} + (1-\alpha) \zeta^{n+\frac{1}{2}} \right) + G_v \left(u^{n+\frac{1}{2}}, v^{n} \right) \right] \\ v^{n+\frac{1}{2}}(k) & = & v^{n}(k) - \Delta t \left[g \frac{\partial}{\partial y} \left(\alpha \zeta^{n} + (1-\alpha) \zeta^{n+\frac{1}{2}} \right) + G_v \left(u^{n+\frac{1}{2}}, v^{n} \right)(k) \right] \\ & & + \Delta t \frac{\partial}{\partial z} \left[k_z \frac{\partial}{\partial z} \left(v^{n+\frac{1}{2}}(k) \right) \right] \end{array} \right.$$

Solve the barotropic tridiagonal linear system dyn2dy_tls_solve

Solve the tridiagonal linear system along z dyn3dy_ztls_solve

$$\left\{ v^{n+\frac{1}{2}}(k) = v^{n+\frac{1}{2}}(k) - \overline{v^{n+\frac{1}{2}}(k)} + \bar{v}^{n+\frac{1}{2}} \right.$$

3D coupling

dyn3d_height_mask

$$\left\{ \begin{array}{rcl} h^{n+\frac{1}{2}} T^{n+\frac{1}{2}} & = & h^{n} T^{n} - \Delta t \left\{ \frac{\partial}{\partial x} \left[\left(\alpha h^{n} + (1-\alpha) h^{n+\frac{1}{2},*} \right) u^{n+\frac{1}{2}} T^{n} \right] \right. \\ & & + \frac{\partial}{\partial y} \left[\left(\alpha h^{n} + (1-\alpha) h^{n+\frac{1}{2},*} \right) v^{n+\frac{1}{2}} T^{n} \right] \\ & & \left. + \frac{\partial}{\partial z} \left[h^{n+\frac{1}{2}} w^{n+\frac{1}{2}} T^{n} \right] \right\} \end{array} \right.$$

Balance between sea surface height and meridional velocity estimates at n+1/2

Time stepping and 3D coupling procedure in dyn3dyx.F90 (5/6)

from dyn3dx

$$\left\{ \bar{u}^{n+1,*} = \bar{u}^{n+\frac{1}{2}} - \Delta t \left[\bar{G}_u \left(u^n, v^n \right) + g \frac{\partial \zeta^{n+\frac{1}{2}}}{\partial x} \right] \right.$$

Time extrapolation, G^n to use old estimate

dyn3dy_tls_coef
dyn3dy_cty
dyn3dy_mtm

$$\left\{ \begin{array}{rcl} \zeta^{n+1,*} & = & \zeta^{n+\frac{1}{2}} - \Delta t \left[\frac{\partial}{\partial x} \left(h^{n+\frac{1}{2}} u^{n+1,*} \right) + \frac{\partial}{\partial y} \left(h^{n+\frac{1}{2}} v^{n+1,*} \right) \right] \\ \bar{v}^{n+1} & = & \bar{v}^{n+\frac{1}{2}} - \Delta t \left[g \frac{\partial}{\partial y} \left(\alpha \zeta^{n+\frac{1}{2}} + (1-\alpha) \zeta^{n+1,*} \right) + \bar{G}_v \left(u^{n+\frac{1}{2}}, v^{n+\frac{1}{2}} \right) \right] \\ v^{n+1}(k) & = & v^{n+\frac{1}{2}}(k) - \Delta t \left[g \frac{\partial}{\partial y} \left(\alpha \zeta^{n+\frac{1}{2}} + (1-\alpha) \zeta^{n+1,*} \right) + G_v \left(u^{n+\frac{1}{2}}, v^{n+\frac{1}{2}} \right)(k) \right] \\ & & + \Delta t \frac{\partial}{\partial z} \left[k_z \frac{\partial}{\partial z} \left(v^{n+1}(k) \right) \right] \end{array} \right.$$

Solve the barotropic tridiagonal linear system dyn2dy_tls_solve

Solve the tridiagonal linear system along z dyn3dy_ztls_solve

$$\left\{ v^{n+1}(k) = v^{n+1}(k) - \overline{v^{n+1}(k)} + \bar{v}^{n+1} \right.$$

3D coupling

dyn3d_height_mask

dyn3dx_tls_coef
dyn3dx_cty

dyn3dx_tls_coef
dyn3dx_mtm

$$\left\{ \begin{array}{rcl} \zeta^{n+1} & = & \zeta^{n+\frac{1}{2}} - \Delta t \left\{ \frac{\partial}{\partial x} \left[\left(\alpha h^{n+\frac{1}{2}} + (1-\alpha) h^{n+1,*} \right) \bar{u}^{n+1} \right] \right. \\ & & \left. + \frac{\partial}{\partial y} \left[\left(\alpha h^{n+\frac{1}{2}} + (1-\alpha) h^{n+1,*} \right) \bar{v}^{n+1} \right] \right\} \\ \bar{u}^{n+1} & = & \bar{u}^{n+\frac{1}{2}} - \Delta t \left[g \frac{\partial}{\partial x} \left(\alpha \zeta^{n+\frac{1}{2}} + (1-\alpha) \zeta^{n+1} \right) + \bar{G}_u \left(u^{n+\frac{1}{2}}, v^{n+1} \right) \right] \\ u^{n+1}(k) & = & u^{n+\frac{1}{2}}(k) - \Delta t \left[g \frac{\partial}{\partial x} \left(\alpha \zeta^{n+\frac{1}{2}} + (1-\alpha) \zeta^{n+1} \right) + G_u \left(u^{n+\frac{1}{2}}, v^{n+1} \right)(k) \right] \\ & & + \Delta t \frac{\partial}{\partial z} \left[k_z \frac{\partial}{\partial z} \left(u^{n+1}(k) \right) \right] \end{array} \right.$$

Solve the barotropic tridiagonal linear system dyn2dx_tls_solve

Solve the tridiagonal linear system along z dyn3dx_ztls_solve

$$\left\{ u^{n+1}(k) = u^{n+1}(k) - \overline{u^{n+1}(k)} + \bar{u}^{n+1} \right.$$

3D coupling

dyn3d_height_mask

$$\left\{ \begin{array}{rcl} h^{n+1} T^{n+1} & = & h^{n+\frac{1}{2}} T^{n+\frac{1}{2}} - \Delta t \left\{ \frac{\partial}{\partial x} \left[\left(\alpha h^{n+\frac{1}{2}} + (1-\alpha) h^{n+1,*} \right) u^{n+1} T^{n+\frac{1}{2}} \right] \right. \\ & & + \frac{\partial}{\partial y} \left[\left(\alpha h^{n+\frac{1}{2}} + (1-\alpha) h^{n+1,*} \right) v^{n+1} T^{n+\frac{1}{2}} \right] \\ & & \left. + \frac{\partial}{\partial z} \left[h^{n+1} w^{n+1} T^{n+\frac{1}{2}} \right] \right\} \end{array} \right.$$

Balance between sea surface height and zonal velocity estimates at n+1

Time stepping and 3D coupling procedure (6/6)

The $G_{u/v}$ is the Right Hand Side of the tridiagonal linear system that corresponds to the explicit part of the scheme. It is estimated in dyn3dx/y_tls_coef.F90. The other coefficients relative to the implicit part of the momentum and continuity equations are also estimated in the same routine : in dyn3dx/y_tls_coef-dyn3dx/y_mtm.h and dyn3dx/y_tls_coef-dyn3dx_cty.h respectively.

$$
\left\{
\begin{array}{rcl}
G_v(u^n,v^n)(k) & = & -f\left(u^n(k)-\overline{u^n(k)}\right) \\
 & & -fu^{n+1/2,(*)} \\
 & & +\pi_y(k) \\
 & & +\frac{1}{\rho}\frac{\partial Pa}{\partial y} \\
 & & +\delta_{kmax,k}\frac{\tau_s}{\rho h^{n,(*)}} \\
 & & -\delta_{1,k}\frac{Cd}{h^{n,(*)}}v^n(1) \\
 & & +\left(u^n(k)\frac{\partial v^n(k)}{\partial x}+v^n(k)\frac{\partial v^n(k)}{\partial y}+w^n(k)\frac{\partial v^n(k)}{\partial z}\right) \\
 & & +\frac{\partial}{\partial x}\left(\nu\frac{\partial v^n(k)}{\partial x}\right)+\frac{\partial}{\partial y}\left(\nu\frac{\partial v^n(k)}{\partial y}\right)
\end{array}
\right.
$$

Remove the barotropic part of Coriolis term at time n, add barotropic part of Coriolis term at time n+1/2

internal pressure gradient

wind and pressure forcing

bottom stress

non-linear advection

viscosity

Rk : n,(*) depending on the step of the estimate

Adaptative time step

The time step remains limited due to non-linear stability reasons, $C = \frac{u_{\max} \Delta t}{\Delta x} \leq C_{crit} \quad avec \quad C_{crit} = 0.7$
but evolves during the simulation according to the hydrological situation.
Typically, time step is larger during neap tides and lower during spring tides.

The algorithm is :

1. $CFL_{ref} = 0$
2. At t, estimation of CFL_{max} at the surface,
3. If $CFL_{max} < CFL_{crit}$, the process goes to 6
4. Else new estimate of dt in order to get $CFL_{max} = 0.5$
5. If dt < 3 s, the runs is stopped
6. $CFL_{ref} = \max (CFL_{max}, CFL_{ref})$
7. If $t < t_{ref} + t_{obs}$, the process starts procedure again from step 2
8. If t > tref + tobs & $CFL_{ref} < 0.84\ CFL_{crit}$, $dt = \min(1.2\ dt, dt_{max})$, the procedure restarts at step 1

The time step is reduced as soon as the CFL criterion is not satisfied.
It is increased after a period (tobs) during which the CFL criterion is satisfied. The time step is limited by dt_{max} and dt_{min}

Numerical schemes

Arakawa C grid (horizontal and vertical)
Finite differences
2nd order spatial centered schemes

$$L(A) = u\frac{\partial A}{\partial x} + v\frac{\partial A}{\partial y} + w^{å}\frac{\partial A}{\partial \sigma}$$

Advection : centered et QUICK
Upstream, QUICK + limiter for tracers

π_x

Internal pressure gradient :
- **along sigma (density jacobian)**
- **density jacobian using cubic spline interpolation (Shchepetkin & McWilliams, 2003)**

$$F_x = \frac{1}{D}\frac{\partial}{\partial x}[D\nu_x\frac{\partial u}{\partial x}] + \frac{1}{D}\frac{\partial}{\partial x}[\nu_x(\frac{\partial H}{\partial x} - \sigma\frac{\partial D}{\partial x})\frac{\partial u}{\partial \sigma}] +$$

$$\frac{1}{D}\frac{\partial}{\partial \sigma}[\nu_x(\frac{\partial H}{\partial x} - \sigma\frac{\partial D}{\partial x})\frac{\partial u}{\partial x}] + \frac{1}{D}\frac{\partial}{\partial \sigma}[\frac{\nu_x}{D}(\frac{\partial H}{\partial x} - \sigma\frac{\partial D}{\partial x})^2\frac{\partial u}{\partial \sigma}]$$

Mellor, 1985 (momentum only)

$$F_x = \frac{\partial}{\partial x}(\nu_x\frac{\partial u}{\partial x}) \qquad F_y = \frac{\partial}{\partial y}(\nu_y\frac{\partial v}{\partial y})$$

ν_x, ν_y

Cst, Smagorinsky, 1963

Modules and functionalities

- **TEST CASES :**
  - Gravity ajustment,
  - seamount,
  - Wetting-drying,
  - channel (tide+river dischage),
  - Unsalty bubble,
  - evaporation / rain,
  - Kelvin wave,
  - Profile
  - Instable barotropic jet
  - Barotropic vortex
  - Baroclinic vortex
  - Advection of spot of tracers
  - conservation of constants in a non-divergent velocity field
  - Deformation of a shape in a non-divergent velocity field

- **PARALLELISATION openMP / MPI**
- **FUNCTIONALITES**
  - Diagnostics
  - Tracking points
  - Passive substances
  - balance, stocks and fluxes calculations
- **MODULES**
  - Trajectories
  - AGRIF package
  - Sedimentology (sand+mud)
  - Biology
  - Contaminant

References

Blumberg AF, Mellor GL. A description of a three dimensional coastal ocean circulation model. In: Three dimensional Coastal Ocean Models. Heaps NS editor, Coastal and Estuarine science, AGU, Washington DC 1987 ; 4 : 1-16.

Lazure P., Dumas F. An external-internal mode coupling for a 3D hydrodynamical model for applications at regional scale (MARS). Advances in Water Resources 2008 (31) 233-250.

Leendertse JJ. A water quality simulation model for well-mixed estuaries and coastal seas. Principles of computations. The rand corporation, RM-6320-RC/1, 1970.

Mellor GL. An equation of state for numerical models of ocean and estuaries. Journal of Atmospheric and Oceanic Technology 1991 ; 8 : 609-611.

Peaceman DW, Rachford H. The numerical solution of parabolic and elliptic differential equations. J. Soc. Ind. Appl. Math. 1995 ; 3 : 28-41

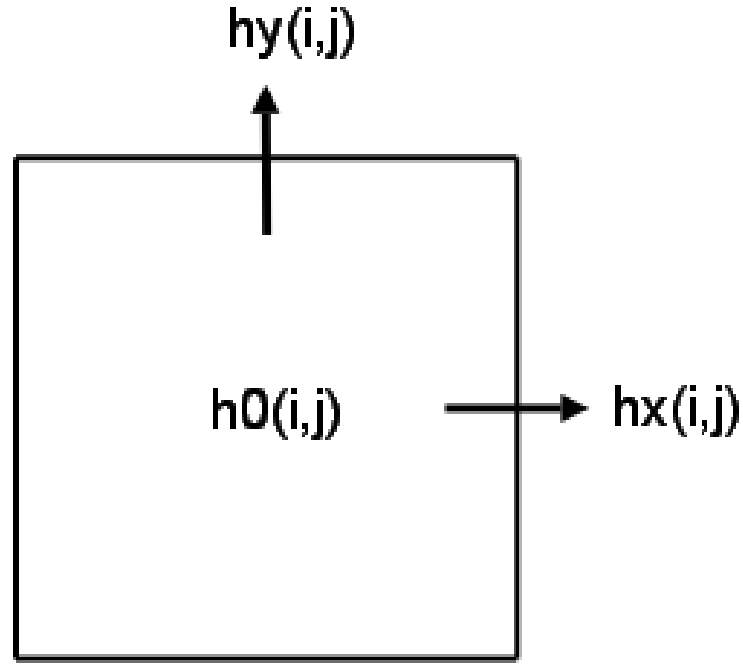
Schwiderski EW. Ocean Tides, Part II : A hydrodynamical interpolation model. Mar. Geodes. 1980 ; 3 : 219-255.

Numerical discretization

MARS variables (1/2)

Metrics

- imin=0 (toujours) : index of the first column
- jmin=0 (toujours) : index of the first row
- imax : index of the last column
- jmax : index of the last row
- kmax : number of layers
  - K=1 bottom layer
  - K=kmax surface layer



Bathymetric variables

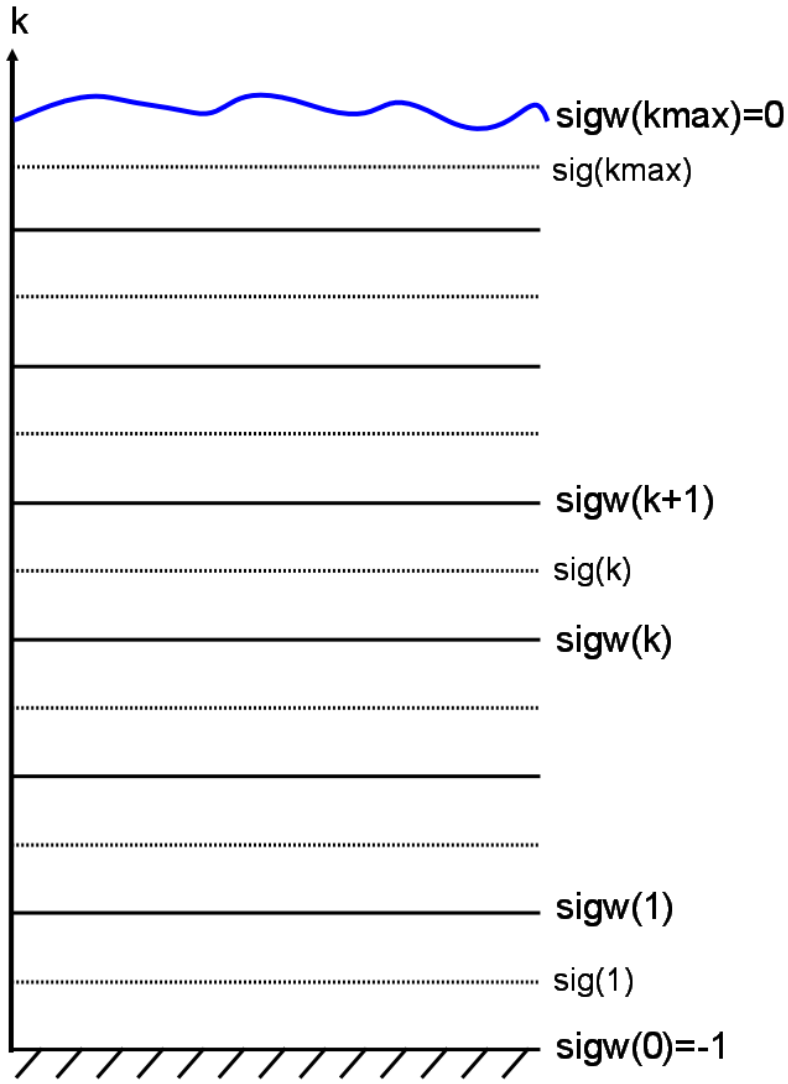
- h0(imin:imax,jmin:jmax)
- hx(imin:imax,jmin:jmax)
- hy(imin:imax,jmin:jmax)

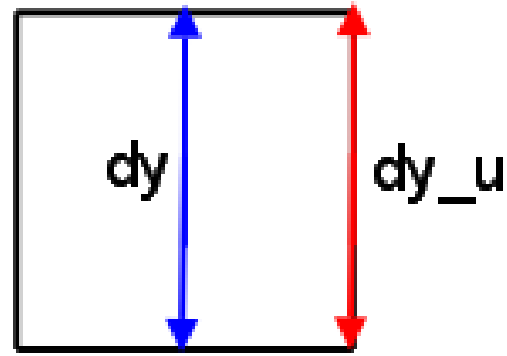
MARS variables (2/2)

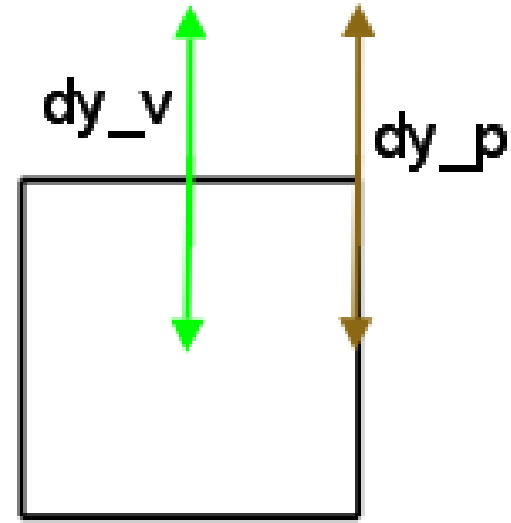
State variables

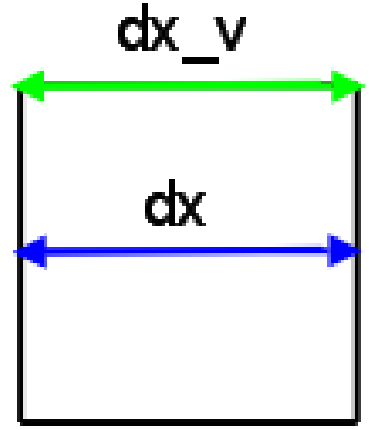
- ssh(imin:imax,jmin:jmax) : sea surface height
- u(imin:imax,jmin:jmax) : barotropic zonal velocity (from row calculation at t+dt/2)
- uz(kmax,imin:imax,jmin:jmax) : 3D zonal velocity (from row calculation at t+dt/2)
- v : barotropic zonal velocity (from column calculation at t+dt/2)
- vz(kmax,imin:imax,jmin:jmax) : 3D meridional velocity (from column calculation at t+dt/2)
- sal(kmax,imin:imax,jmin:jmax) : salinity
- temp(kmax,imin:imax,jmin:jmax) : temperature
- cv_wat (nb_var,kmax,imin:imax,jmin:jmax) : substance

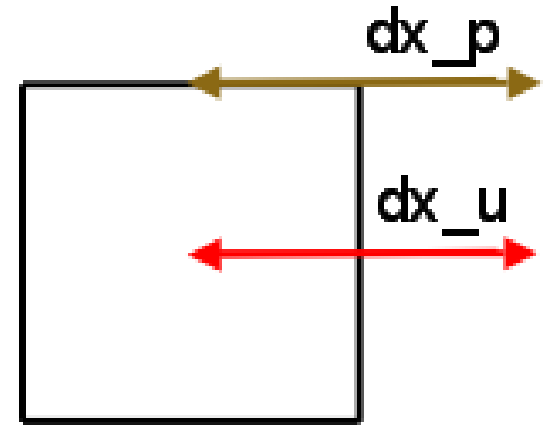
MARS grid (1/2)


k
sigw(kmax)=0
sig(kmax)
sigw(k+1)
sig(k)
sigw(k)
sigw(1)
sig(1)
sigw(0)=-1

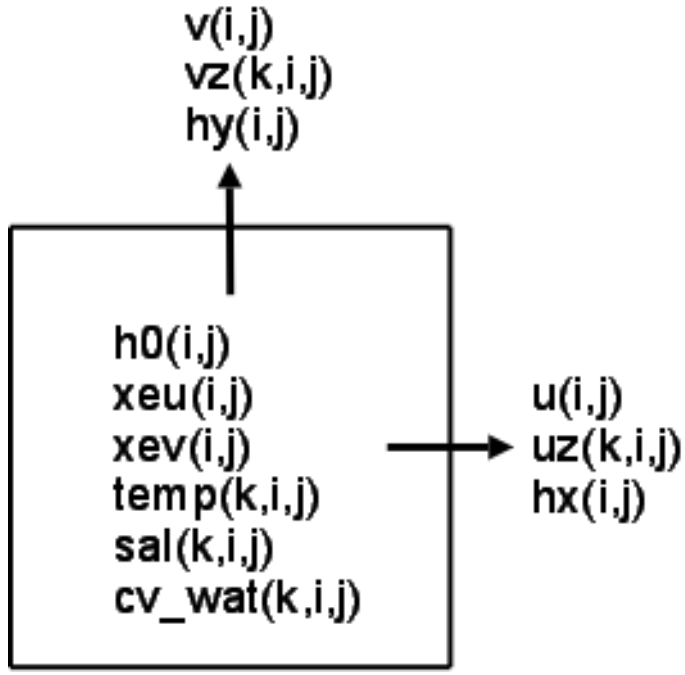

dy
dy_u

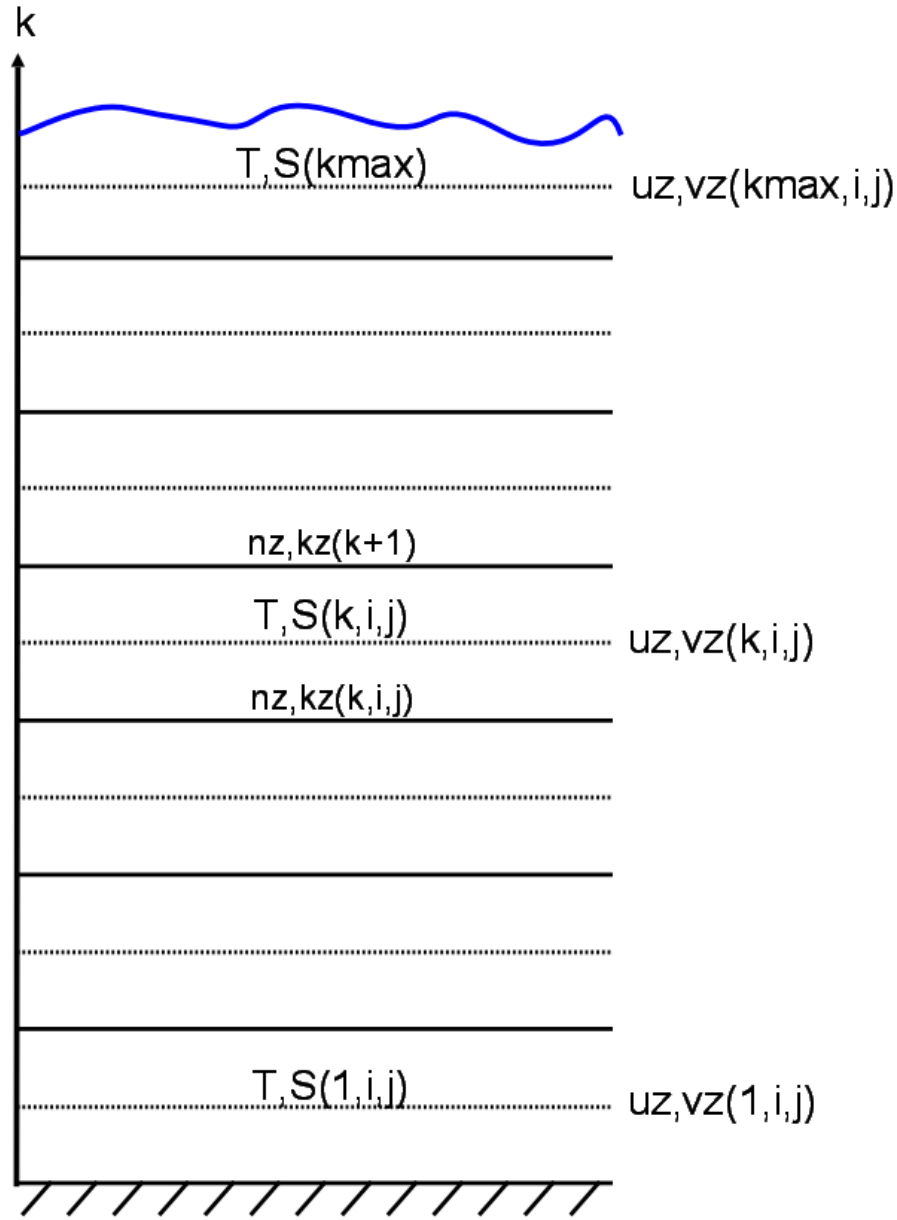

dy_v
dy_p


dx_v
dx


dx_p
dx_u

Spatial discretization (2/2)

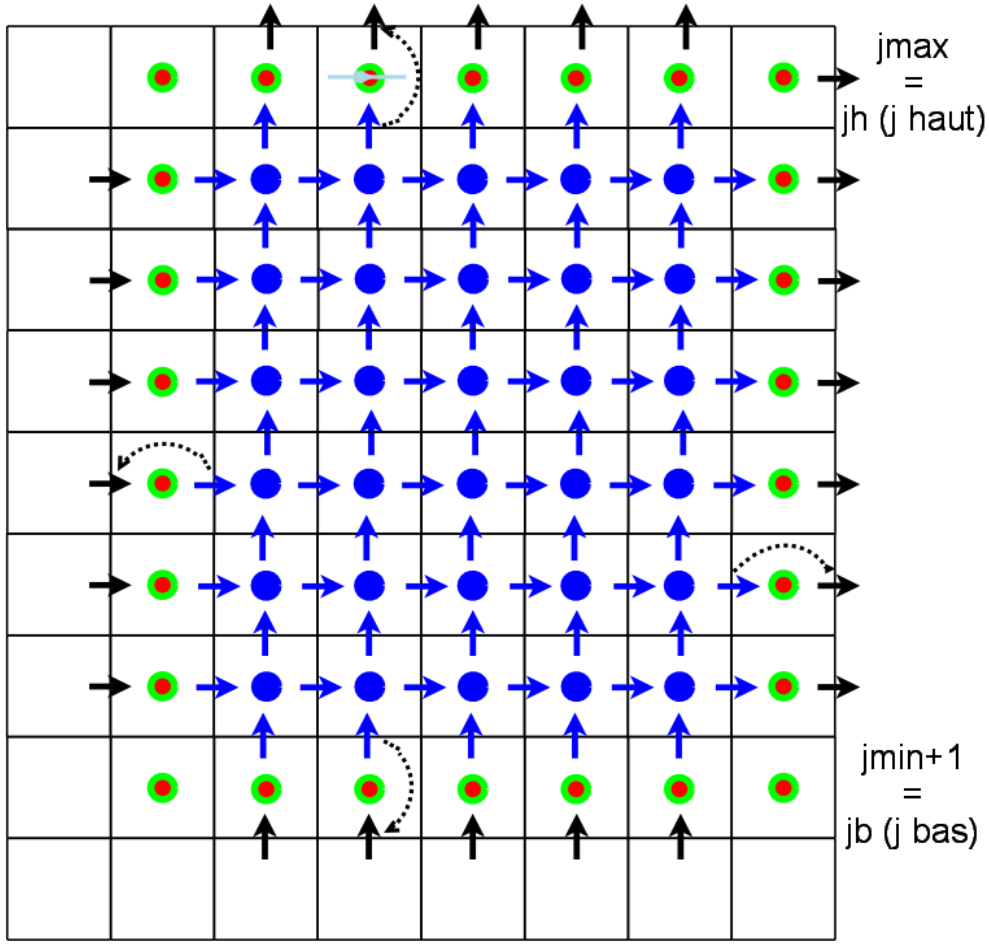




Open boundaries

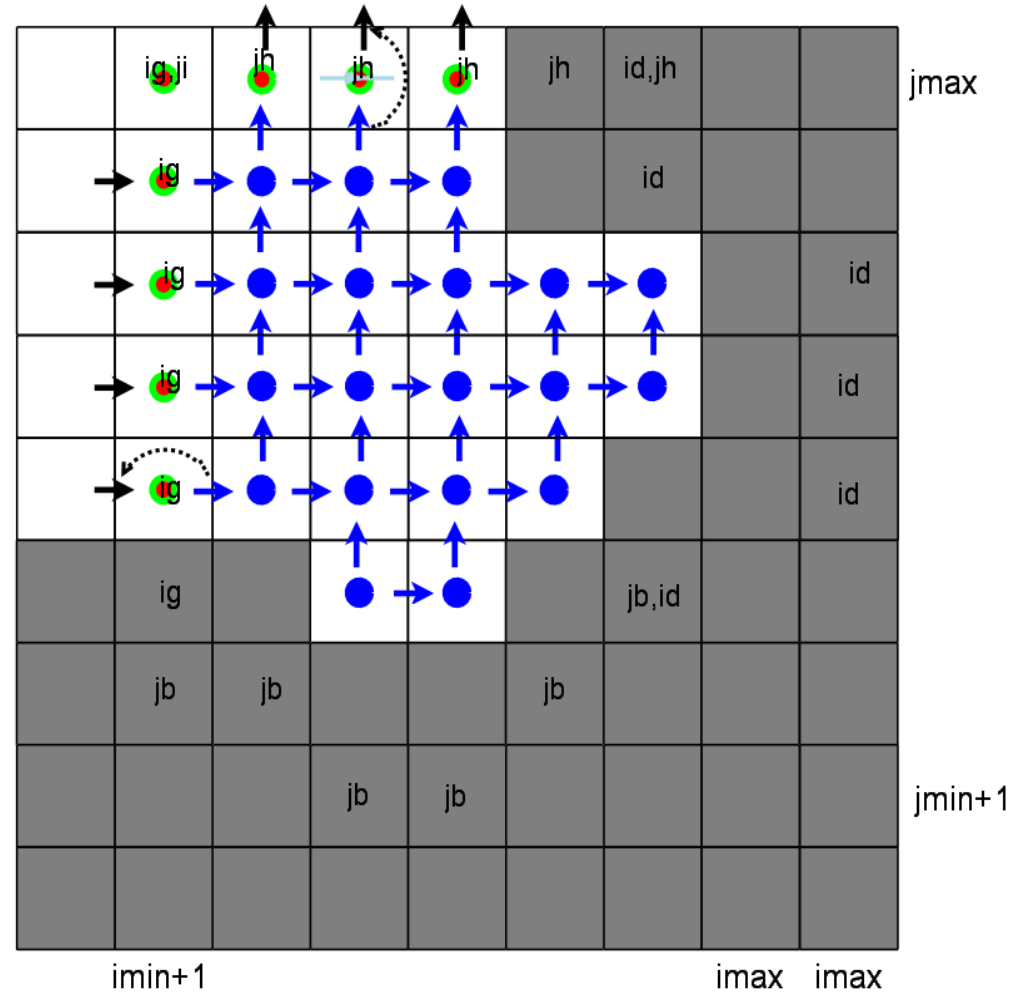
Domain with four open boundaries

XE, u, v, T and S are calculated
XE is prescribed
T and S are prescribed or interpolated (upwind scheme)
u,v are duplicated (key_obc_mars)

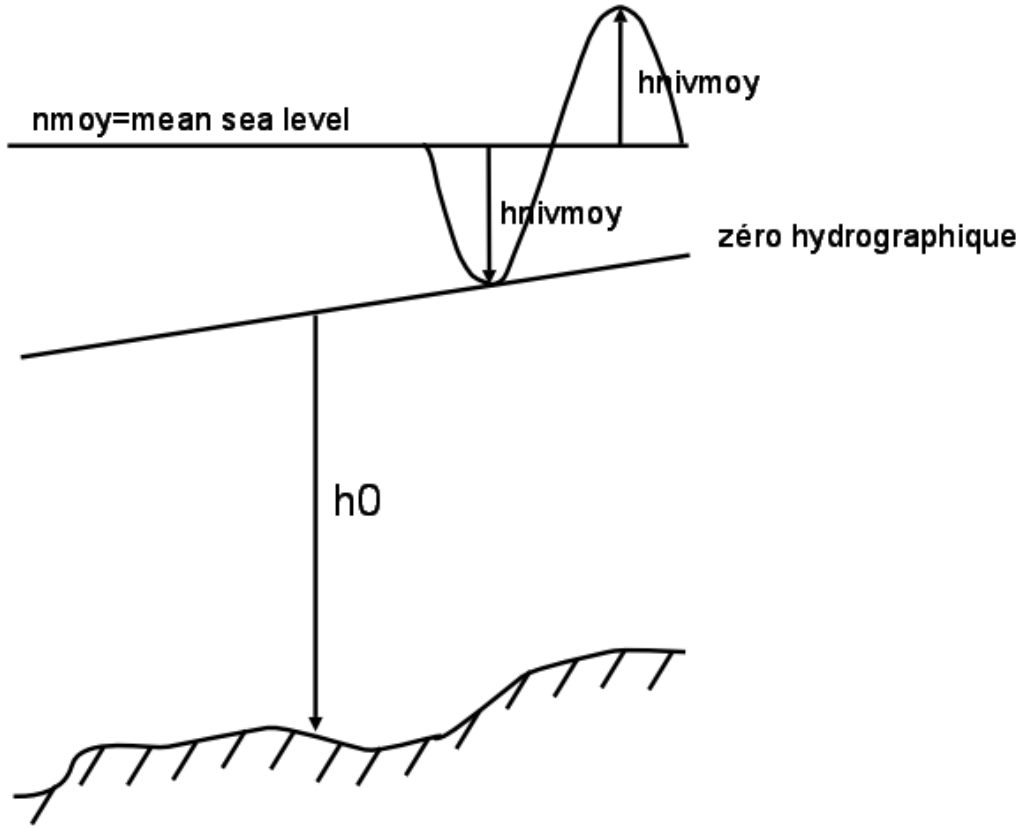


Domain with two open boundaries

XE, u, v, T and S are calculated
XE is prescribed
T and S are prescribed or interpolated (upwind scheme)
u,v are duplicated (key_obc_mars)



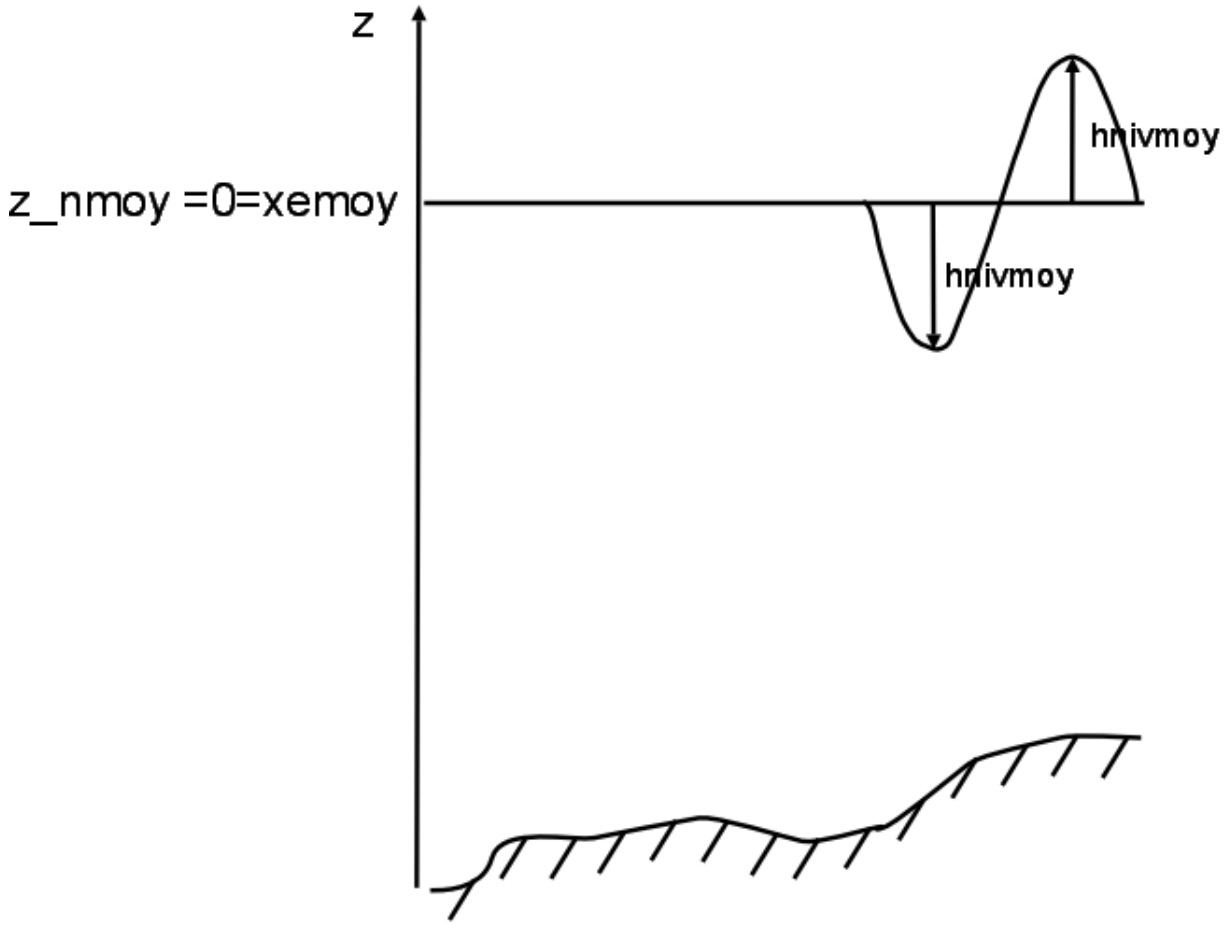
Parametrization of the bathymetry (1/4)



hminim marks off the coastline because it is the minimum value of the bathymetry (hx,hy)
hminim = - max (|hnivmoy|) along the coastline

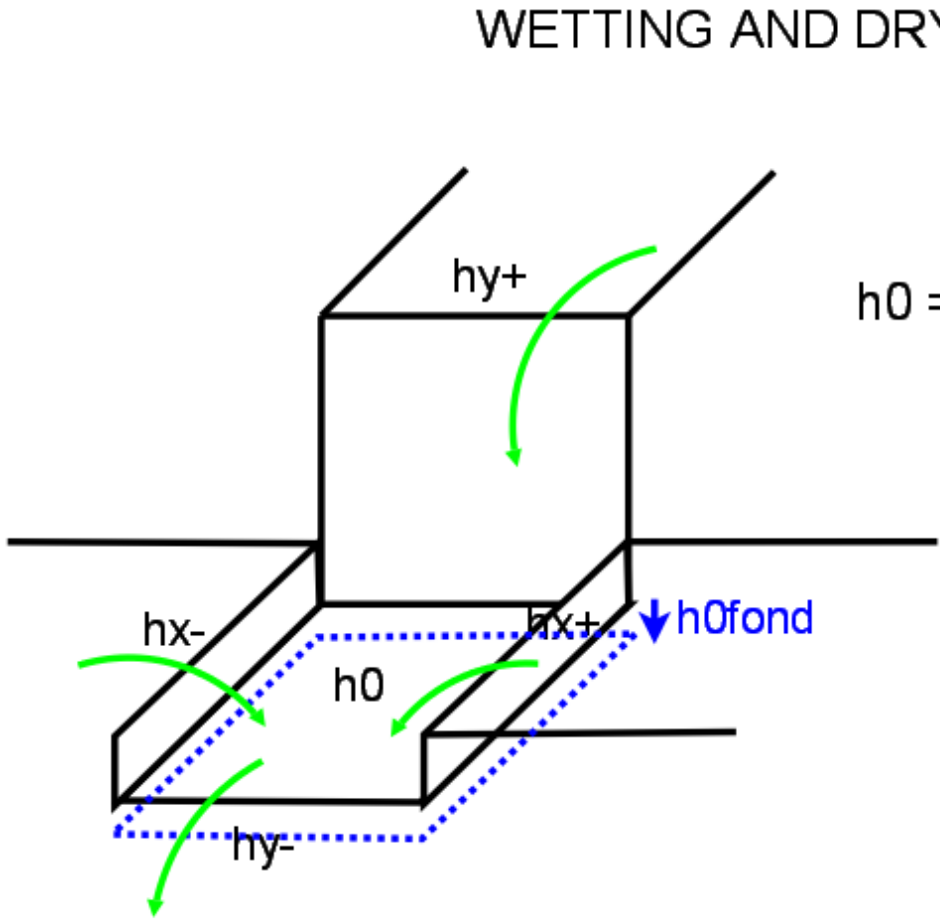
In microtidal area, if the reference for the bathymetry is the level of lowest astronomical tides, the shift hnivmoy is applied to the initial bathymetric value. Then, the tidal signal evolves around the mean sea level. All along the coast, hminim is equal to hnivmoy and marks the coast line

Parametrization of the bathymetry (2/4)



The parameter xemoy may be used when the model vertical reference of the bathymetry derives from a preexisting level network (IGN 69, NGF…) that differs from the hydrographic zero (level of lowest astronomical tides)

Parametrization of the bathymetry (3/4)

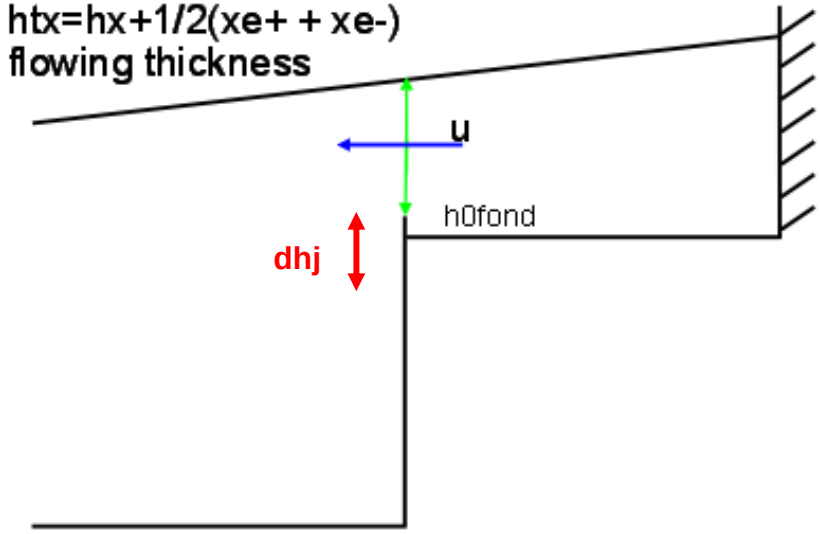


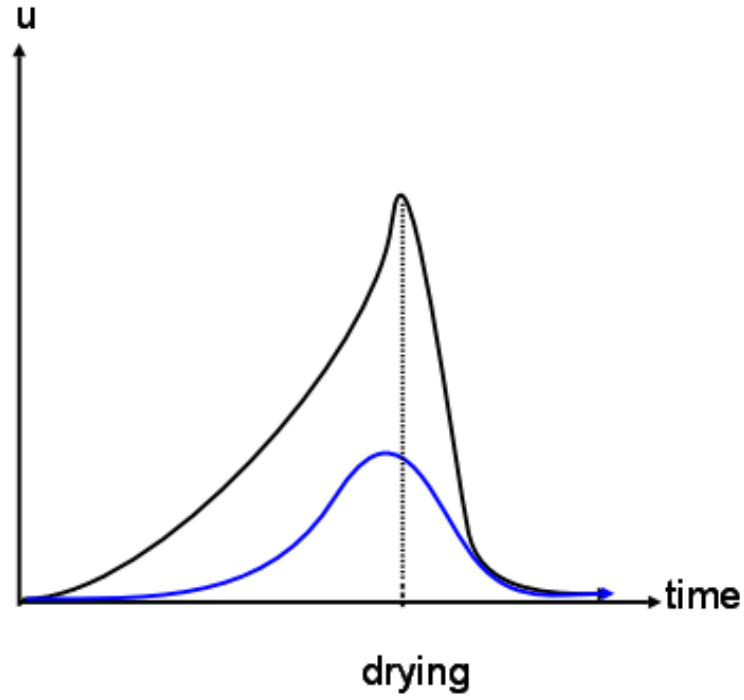
h0 = max(hx-,hx+,hy-,hy+)

h0fond is a residual thickness used to ensure h0+xe keeps positive.
As the continuity scheme is not positive definite, it might be possible to export more water mass than available in the cell during the drying.

Parametrization of the bathymetry (4/4)

WETTING AND DRYING





drying

dhj is a fictive thickness of the flow, added to the water height to avoid overshoots of velocities during drying

Parametrization of vertical mixing (1/2)

$$\frac{1}{D}\frac{\partial(\frac{nz}{D}\frac{\partial u}{\partial \sigma})}{\partial \sigma} \qquad nz, kz$$

Turbulence models

0 equations : turb_nbeq=0

turb_0eq_option=1 cst

turb_0eq_option=2 Prandt model $\quad nz = u^* H / 15\sigma\sqrt{(1-\sigma)} \quad ; \quad u^* = 0.4 / \log(\sigma H / z_0) u_{bottom}$

turb_0eq_option=3 Quetin model $\quad nz = l^2 \frac{\partial u}{\partial z} \quad ; \quad kz = f(Ri) l^2 \frac{\partial u}{\partial z} \quad ; l \, analytical$

turb_0eq_option=4 Pacanovski et Philander, 1981

$$nz = \frac{\nu_0}{(1+\alpha Ri)^n} + \nu_b \quad ; \quad kz = \frac{\nu}{(1+\alpha Ri)} + k_b$$

$$Ri = \frac{\partial b}{\partial z} / \left|\frac{\partial U}{\partial z}\right|^2$$

1 equation : Gaspard et al, 1990

turb_nbeq=1

$$nz, kz = f(ect, l) \quad ; \quad ect \, computed \; ; \quad l \, evaluated$$

Parametrization of vertical mixing (2/2)

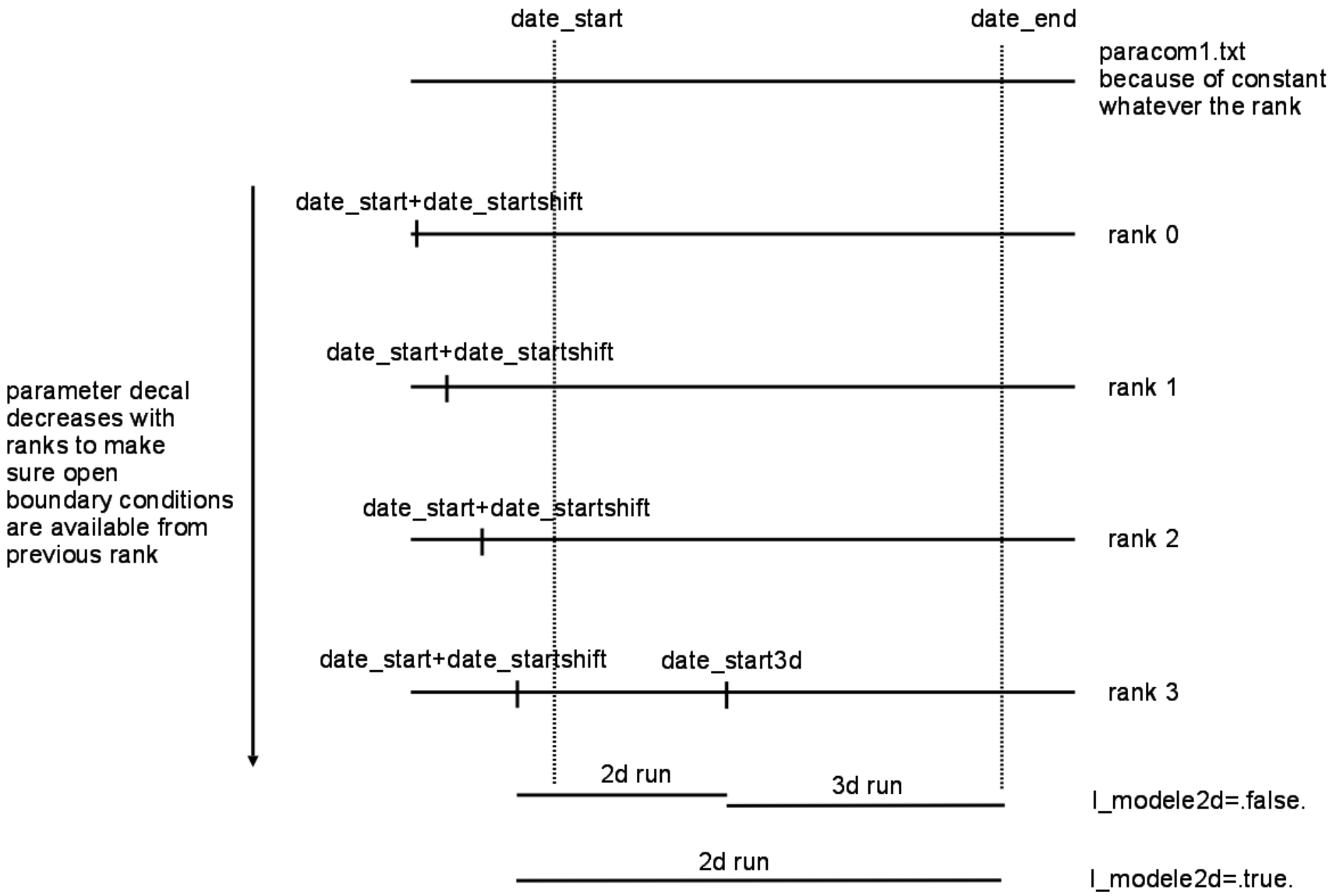
$$\frac{1}{D}\frac{\partial(\frac{nz}{D}\frac{\partial u}{\partial \sigma})}{\partial \sigma} \qquad nz, kz$$

Turbulence models

2 equations : turb_nbeq=2 $\quad nz, kz = f(k, kl, l) \quad ; \quad k \text{ et } kl \text{ computed} \; ; \quad l \text{ evaluated}$

turb_2eq_option=1	k-kl model
turb_2eq_option=2	k-epsilon model
turb_2eq_option=3	k-omega
turb_2eq_option=4	generic length scale model

Parametrization of the run period



Structure of the code (1/11)

Parameters.F90

Parametrization of the configuration
imax,jmax,kmax
nrac,no

↓

Main.F90

Allocation of variables and initialization

↓

Step.F90

Forcings + open boundary conditions,
Surface boundary conditions…

↓

step3dxy.F90 **(computation along the row)**
following by
step3dyx.F90 **(computation along the column)**

Hydrodynamics
+
Tracers

Structure of the code : main.F90 (2/11)

call **init**	! Read namelist paramain.txt, paracom.txt, paraspec.txt,… ! Initialize a few variables and do some tests on cpp keys…
call **bathy**	! Read (and modify) the bathymetry
call **grid_metric**	! Initialize the grid
call **sub_read_var** call **obc_subsref**	! Read variables.dat file describing substances ! Allocate substance variables ! Make variables in order ! Initialize open boundary arrays for substance
call **sed_alloc**	! Allocate sediment variables

if (l_repsed) then
call **sed_init_fromfile**
else
call **sed_init**
endif
	! from file
	
	! Variable.dat

call **sed_settlveloc_ini** If(l_morphocoupl) call **sed_morphoninit**	! Initialize the settling velocity ! Initalize for morphology
call **sub_read_vardiag**	! Read vardiag.dat file describing diagnoses ! Allocate diagnoses variables (only for module biology)

Structure of the code : main.F90 (3/11)

call **init_xe**(ssh**)**	! Initialize the sea surface height
Call **init_height**(ssh)	! Initialize heigt variables for vertical integration
call **init_uvz**(uz,vz)	! Initialize the 3d velocities
call **init_uv**(u,v,uz,vz)	! Initialize the barotropic velocities
call **init_viscoef**	! Initialize viscosity coefficients
call **init_turbcoef**(ect,psi)	! Initialize some turbulent coefficients
call **init_tssub(**sal,temp,cv_wat**)**	! Initialize sal, temp, cv_wat
call **obc_tsref**	! set T and S open boundary condition to reference values
if (l_initfromfile)	
call **init_fromfile**	**!** Initialize ssh,uz,vz,u,v,sal,temp from a file
call **biolo_userinit(**cv_wat**)**	! Special initialiszarion of biological variables
call **conta_read_react**	! Reading reactions between contaminant variables
call **user_init_vdriv**	**!** Initalization of driving variables
call **output_init**	! Read output.dat file and initialize output parameters
call **sub_budget_readdomain(ssh)**	! If (_bilan) initialize zones for budgets and fluxes computations
call **summary**	**!** Summarize initialization part
call **meteo**	**!** Set meteorological fluxes if l_meteo_stat=T,l_meteo_hom=T
call **river**(ssh)	**!** Read **river.dat** , init….
call **outflow**	**!** Read **outflow.dat** file, describing outflows
	! initialize outflow inputs with time interpolation

Structure of the code : main.F90 (4/11)

call **sub_flx_read**	! If temporal variations in cv_rain or in atm.fluxes
if(icon >=1)call **ts_stateeq_*(**sal,temp,bz**)**	**!** Estimate buoyancy from sal and temp
if(**l_points**) call **track_points(**u,v,uz,vz,ssh,sal,temp,cv_wat,....)	! Initialize variables for traking
if(**l_diag**) call **diagnostic(**u,v,uz,vz,ssh,sal,temp,cv_wat,ect,bz**)**	! Initialize diagnoses variables
call **output_mng**	! manage outputs files
if(**l_demer(nb)**) call **out_demerliac**(ssh,u,uz,v,vz,sal,temp,cv_wat,....)	! Initialize demerliac filtering
call **traj_init2d** or **traj_init3d** call **ibm_init**	! Initialize the trajectory part ! Initalize IBM

Structure of the code : main.F90 (5/11)

do while(t <= tfin .or. mod(nbouc,2)==0) call step()	! **Time integration**
end do	
Call dealloc End program	

Structure of the code : step.F90 (6/11)

mod(nbouc,2)==0
  if(l_saverestart_1file.or.l_saverestart_bydate)
  call **out_saverestart**(…)
  call **sed_outsaverestart**
	odd nbouc
	! Save restart file

t=t+0.5 dt ; nbouc=nbouc+1	! increment the time
if(l_genpot) call **tide_astroarg** if(l_genpot) call **tide_genpot**	! Tide potential

call **meteo**
 	
if(.not.l_meteo_stat .and. .not.l_meteo_hom)
call **sflx_read**
 	
call **sflx_flux(temp)**
 	
if(nbfl > 0) call **river**
	! Fluxes at t
	! if l_meteo_stat or l_meteo_hom
	
	! Fluxes at t
	
	! Exchanges at the air-sea interface
	
	! Input of river discharge at t

call **sub_sflx_read** call **user_ftime_vdriv** call **user_vintermediate**	! Interpolation date for cvrain and flxatm ! Define driving variable at t ! Computing intermediate variables
call **dyn3d_dt**(uz,vz)	! CFL criterion, ! increase or decrease time step

Structure of the code : step.F90 (7/11)

```
If (t .ge. tdebsubs) then                                   ! Module sedimentology
  call sed_skinstress
  call sed_erosion
  call sed_settlveloc                                       ! Estimate of settling velocity
  if (isand >= 1) call sed_sandconcextrap
  call sed_depflx
endif

call conta_reac_after (cv_wat,sal,temp,ssh)                 !reactions at equilibrium (contaminant)

! First half time-step (mod(nbouc,2)==1)

call step3d(2d)xy                                           ! Estimate ssh,u,uz,v,vz,sal,temp
                                                            ! u uz are intermediate estimate in 3d
If (l_diag)
call diagnostic(u,v,uz,vz,ssh,sal,temp,cv_wat,ect,bz)

If (t .ge. tdebsubs)
 call sed_effdep(ssh)                                       ! module sedimentology
 call sed_consol_diffu

call sub_budget_out                                         ! Budget output

If (l_points)
call track_points(u,v,uz,vz,ssh,sal,temp,cv_wat,ect,bz)
```

Structure of the code : step.F90 (8/11)

! **Second half time-step** (mod(nbouc,2)==2) call step3d(2d)yx	! Estimate ssh,u,uz,v,vz,sal,temp ! v,vz are intermediate estimate in 3d
If (l_diag) call **diagnostic**(u,v,uz,vz,ssh,sal,temp,cv_wat,ect,bz)	

If (t .ge. tdebsubs) then
call **sed_effdep**(ssh)
call **sed_consol_diffu**
if (l_morphocoupl) call **sed_morpho**
Endif
	! **module sedimentology**

if (l_bilan) call **sub_budget_out**(t,cv_wat,cv_sed,ssh)
 	
If (l_points)
call **track_points**(u,v,uz,vz,ssh,sal,temp,cv_wat,ect,bz)	

call **output_mng**	! manage outputs files, save data output if required ! Estimate of maximum velocities

Structure of the code : step3dxy.F90 (9/11)

	! **Initialize xeuv, uv and uv2**
call **tide_harmcompo** or **obc_2D_rank…** or **obc2d_cas**(xeuv) (! if test case) If (l_obc_ogcm) call **obc_ogcm** call **obc_subs** call **obc_combine**(xeuv) fields call **obc_apply_ssh**(xeuv)	! **Open Boundaries** ! Prescribe ssh at open boundaries ! Read ogcm open boundary conditions ! Obc for substances ! Combine tide and ocgm open boundary ! Apply ssh at the open boundaries
call **dyn3dxy**(u,v,ssh,uv,uv2,xeuv,uz,vz,bz) call **dyn3d_wz**(ssh,xeuv,uz,vz)	! **Dynamic** ! Estimate new xeuv,uv,uvz at t+dt/2 ! Estimate vertical velocity wz
call **init_flux**	! Initalize phicon and phieau
if (t .ge. tdebsubs) call **biolo_dyn_zwat(**ssh,cv_wat,sal,temp**)** **call radz_reduce**	! **module Biology** ! Computation of inter-cell influences (extinction coefficient…) **! module Contaminant**
if(nbrej > 0) call **outflow**	**! module Tracer** ! Prescribe discharges at t+dt/2

Structure of the code : step3dxy.F90 (10/11)

if(t .ge. tdeqsubs) then CALL **biolo_sksc_wat**(cv_wat,sal,temp,ssh) CALL **biolo_sksc_sed**	! **module Biology** ! Transformations inside cell of water and sediment ! Estimate new concentrations in sediment cv_sed(1:nv_state,:,:,:)
CALL **conta_sksc_wat**(cv_wat,sal,temp,ssh) endif	! **module Contaminant**
call **sflx_surf**(temp,sal,ssh)	! Estimates **atmospherical fluxes**, ! temperature and salinity fluxes
call **tssub_eq**(xeadv,ssh,sal,temp,cv_wat)	**transport equation resolution** - ! Dissolved Tracers estimate –
If(l_bilan .and.t > tdebsubs) call **sub_budget_flxcum(cv_wat,ssh)**	! Cumulating dissolved substances fluxes for budget
Do l=1,nb_ws_max call **parsub_eq**(…) Enddo	! Particulate Tracers estimate ! By group of substances (same settling velocity)
call **ts_obc3d_apply**(uz,vz,sal,temp) call **sub_obc3d_apply**(uz,vz,cv_wat)	! **Apply boundaries limits conditions** ! Apply obc for T and S ! Apply obc for cv_wat (dissolved)

Structure of the code step3dxy.F90 (11/11)

call **ts_stateeq_***	! Buoyancy from state eq
call **turb***	! Estimate mixing coeficients
call **traj_3d** (ssh,uz,vz)	! **Module trajectories (3D)**
call **ibm_3d** (ssh,uz,vz,sal,temp,cv_wat,boz)	! **Module IBM**

As step3dxy

How to set up a new configuration ?

- **cd $UDIR/CONF/CONF-CASE** (*see also envt_manual.ppt*)
  1. Create the configuration's directories (mkconfdir)
  2. Set up the code (gmake install)
  3. Define the grid size (getfile WORK/parameters.F90[rank*])
     - ✓INC/parameters.F90[_rank*]
     - ✓define imin=0, jmin=0,imax=,jmax=,kmax=,no=,nrac= (same as configuration name usually)
  4. compile

- **cd $RDIR/CONF/inputs**
  1. Head.CONF (keep exactly the same format !)
  2. Bathymetric files (hxhy or netcdf files (file_bathy=); nmoy (file_bathy_meanlev=))
  3. Tide files (l_tide_harmcompo=T)
  4. Atmospherical file (l_file_meteo=) or academic tests (l_meteo_stat=T, l_meteo_hom=T, wind_veloc= ; l_sflx_solarcst=T, sflx_solarcst=)

- **cd $RDIR/CONF/CONF-CASE**
  1. Set your parametrization up
  2. run

realistic configurations : rank management (1/3)

Return to Summary

(see also envt_manual.ppt)

- **Compilation of each rank**
  - cd $UDIR/CONF/CONF-CASE
  - vi **makefile**, RANK = 0
  - **gmake clean** ; **gmake**
  - vi makefile, RANK = 1
  - gmake clean ; gmake
  - ***

- **Parametrization of namelists**
  - **cd $RDIR/CONF/CONF-CASE**
  - **paracom.txt and paramain.txt are the same whatever the rank**
  - **paraspect.txt changes for each rank**
    - ✓*date_startshift (if negative, date_startshift decreases as the number of the rank increases)*
    - ✓*l_tide_harmcompo=.true. Only for rank 0*
    - ✓*l_bathy_meanlev=.true. Except for the rank 0 (because taken into account in the bathymetry file)*

realistic configurations : rank management (2/3)

Example of differences between two paraspec.txt : one for rank 0, the other one for rank 1

- **Rang 0**
  - date_startshift=5
  - file_bathy='../../inputs/hxhy. bzhk0'
  - l_bathy_meanlev=.false.
  - file_bathy_meanlev='../../inputs/nmoy. bzhk0'
  - dtini=400.d0
  - dtmax=1000.d0
  - l_tide_harmcompo=.true.

- **Rang 1**
  - date_startshift=4
  - file_bathy='../../inputs/hxhy.bzhk1'
  - l_bathy_meanlev=.true.
  - file_bathy_meanlev='../../inputs/nmoy.bzhk1'
  - dtini=200.d0
  - dtmax=400.d0
  - l_tide_harmcompo=.false.

realistic configurations : paraspec.txt : open boundary management (3/3)

- Look at your domain, and specify where are the open boundaries
  - l_obc_east=.true./.false. (if open/closed boundary)
  - l_obc_west=….
- Specify the file containing the obc fields issued from an ogcm and extract executable
  - file_obc_n='../../inputs/obc_north.nc'
  - file_obc_e=…
- Which ogcm variables do you want to read and use at open boundaries ? Specify :
  - l_obc_ogcm_rssh=.false. ! Read the sea surface height
  - l_obc_ogcm_rs=.false. ! Read the salinity
  - l_obc_ogcm_rt=.false. ! Read the temperature
  - l_obc_ogcm_ruv=.false. ! Read the barotropic velocities
  - l_obc_ogcm_ruvz=.false. ! Read the 3d velocities
- To relax the salinity and temperature towards the ogcm values over a bandwidth specified by obc_wid
  - l_obc_ogcm_relax=.true.
  - obc_coef_relax=0.7e-06 (time relaxation in s^{-1})
  - dtmax=1000.d0
  - l_tide_harmcompo=.true.
- What kind of physics do you want to introduce through the open boundaries ?
  - l_obc_ogcm=.true. ! If nested with an ogcm
  - l_obc_tide=.true. ! If tide and nested with a previous rank
- Which operator do you want to use ? Choose one of the following items
  - l_obc_diri=.false. ! Dirichlet
  - l_obc_mars ! Zero gradient of velocities
  - l_obc_char ! Characteristics method

Namelist variables

- ✓ presented grouped by themes
- ✓ **List of parameters read in namelist is available in a separate doc (list_param_namelist.doc) *(available later)***

Namelist variables grouped by themes

- ✓ ADVECTION OF TRACERS
- ✓ AMOSPHERIC FIELDS
- ✓ BATHYMETRIE
- ✓ DATE
- ✓ DIAGNOSTICS
- ✓ DIFFUSION
- ✓ ECOSYSTEM
- ✓ GENERAL
- ✓ EQUATION OF MOVEMENT
- ✓ NETCDF REFERENCE
- ✓ OFFLINE
- ✓ OPEN BOUNDARY
- ✓ OUTPUT
- ✓ RADIATIVE HEAT FLUX
- ✓ RESTART

- ✓ RIVER, OUTFLOW
- ✓ SAVING
- ✓ SIGMA
- ✓ STATE EQUATION
- ✓ THERMODYNAMICS
- ✓ TIDE
- ✓ TIME INTERPOLATION
- ✓ TIME STEPS
- ✓ TRAJECTORIES
- ✓ TURBULENCE / VERTICAL MIXING
- ✓ VISCOSITY
- ✓ WIND FIELD

- ✓ Open Boundaries examples

namelist variables
1st configuration

(M) : the variable is inside paramain.txt
(C) : the variable is inside paracom.txt
(S) : the variable is inside paraspec.txt

DATES			
namelists	**variables**	**cpp key**	**observation**
nmlmain (M)	date_ref = '01-Jan-1900 00:00:00‘		Do not change
namdate (C)	date_start = '06/03/2005 00:00:00' datefin = '09/03/2005 00:00:00'		*simulation starting date* *simulation fisnishing date*
namdate3d (S)	date_start3d = '03/02/2001 12:00:00‘ date_startagrif = '25/01/1998 11:12:00' date_startshift = 0.0		*3D run starting date (== or > date_start)* *AGRIF zoom starting date* Rank management

Return to Summary

namelist variables
1st configuration

Return to list of namelist variables

(M) : the variable is inside paramain.txt
(C) : the variable is inside paracom.txt
(S) : the variable is inside paraspec.txt

GENERAL			
namelists	**variables**	**cpp key**	**observation**
nmlmain (M)	comdir = '../' jhemisph=1		./ without rank ../ if ranks 1/-1 in North/South hemisphere
namhead (C)	file_head = '../../inputs/head.cool' confname_mother = 'cool' confname = 'cool' suffix = 'V9.06'		
namriv (S)	icon = 2		/=2 to make tests on buoyancy impact 3D
namparanum (S)	l_modele2d=.true. cflcrt=0.6		If set to .true., 2D run even though kmax>1

Return to Summary

namelist variables
1st configuration

Return to list of namelist variables

(M) : the variable is inside paramain.txt
(C) : the variable is inside paracom.txt
(S) : the variable is inside paraspec.txt

SIGMA			
namelists	**variables**	**cpp key**	**observation**
namsigma (S)	l_equisig=.true. sig(1)= 0.05 hc = 5.0 (meters) B_sig Theta_sig L_grid_angle l_corio_cst=.false.	 0 < Bottom control < 1 0 < Surface control < 20 key_grid_rotated	3D 3D Generalized sigma Generalized sigma Generalized sigma

TIME STEPS			
namelists	**variables**	**cpp key**	**observation**
namparanum (S)	dtini=400.d0 dtmin=200.d0 dtmax=800.d0		dtmin<dt<dtmax

namelist variables
1st configuration

BATHYMETRY			
namelists	**Variables**	**cpp key**	**observation**
nambathy (S)	file_bathy='../../inputs/hxhy.cool0' file_bathy_meanlev='../../inputs/nmoy.cool0' name_bathy_hx_b='HX',name_bathy_hy_b, name_bathy_h0_b l_bathy_meanlev=.false. name_bathy_meanlev='nmoy' mslshift=0.0 bathy_nbsmooth=0 l_bathy_comb=.false. l_bathy_pit=.false. hminim=-9.9 hrdef=20.0 h0fond=0.01 dhj=0.05 hminfrot=0.5 hminkxky=0.5 l_smalldepth=.true. hm=1.0 l_bathy_save=.false.	key_netcdf Key_netcdf	should not be changed

namelist variables
1st configuration

(M) : the variable is inside paramain.txt
(C) : the variable is inside paracom.txt
(S) : the variable is inside paraspec.txt

RESTART			
namelists	**variables**	**cpp key**	**observation**
nmlreprise (M)	name_in_dt = 'DT' name_in_xe = 'XE' …		
mamreprise (S)	l_initfromfile = .false. file_init = 'save.nc' l_init_restart=.false. l_init_rtime=.false. l_init_rdt=.false. l_init_rssh=.false. l_init_rbtvel=.false. l_init_r3dvel=.false. l_init_rwz=.false l_init_rturb=.false. l_init_rsal=.false. l_init_rtemp=.false.		Set all the boolean to .true. to get exact results when restarting from a restart file after a break in the simulation. To read an input file prepared from extract program, specify which variables you want to read and define l_init_restart=.false.

namelist variables
1st configuration

Return to list of namelist variables

(M) : the variable is inside paramain.txt
(C) : the variable is inside paracom.txt
(S) : the variable is inside paraspec.txt

OUPUT			
namelists	**variables**	**cpp key**	**observation**
nmlsortie (M)	l_maxu=.true. name_out_h0='h0' … riog_valid_min/max		Maximum velocities are printed on the screen (iscreen=6) or in the listing file (iscreen=12) Range of valid value for the variable
namsortie (S)	file_output= './output.dat' iscreen=12 iscreenlog=18 iwarnlog=16 ierrorlog=17 l_out_nc4par l_out_pack	-Dkey_MPI_2D	Get it empty The simulation is stopped if an error occurs. The error si described in file error_suffix.log True : each cpu writes its local domain in a unique global file True : the data values are packed, i.e. saved in short precision Packed_value = NINT((value-offset)/scale_factor)

Return to Summary

namelist variables
1st configuration

Return to list of namelist variables

(M) : the variable is inside paramain.txt
(C) : the variable is inside paracom.txt
(S) : the variable is inside paraspec.txt

SAVING			
namelists	**variables**	**cpp key**	**observation**
namsauv (S)	l_saverestart_1file = .false. l_saverestart_bydate = .false. file_saverestart= 'save.nc' saverestart_step = 1.0d0		

namelist variables : 1st configuration

OPEN BOUNDARY CONDITIONS

namelists	variables	cpp key	observation
namobc (S)	l_obc_ogcm_rssh=.false. l_obc_ogcm_rt=.false. l_obc_ogcm_rs=.false. l_obc_ogcm_rc=.false. l_obc_ogcm_ruv=.false. l_obc_ogcm_ruvz=.false. l_obc_cyc=.true. l_obc_cycl_x/y=.false. l_obc_mars=.true. l_obc_char=.true. l_obc_diri=.true. l_obc_tide=.true. L_obc_ogcm.true. l_tide_harmcompo=.true. l_obc2drank_save=.true. l_obc_south=.false. file_obc_s='./' … l_obc_relax=.false. obc_coefrel=0.7e-05	key_tide_fes2004 or…	Reading of data issued from another MARS configuration or another model. Data are spatially interpolated off-line with the use of the program named extract. Data are interpolated on time during the run l_obc_rssh=.true., the ssh relative to the mean circulation comes from a coarser model If periodic zonal/meridional periodicity No gradient of velocity at open boundary Characteristics method Dirichlet OBCs come from a previous rank OBCs come from an OGCM Saving of open boundary condition (rank)

namelist variables : 1st configuration
OBC 2D – BISC rang 0

Return to list of namelist variables

Type of dynamics	Where	Type of forcing	Variable	Operator
Tide dynamics	l_obc_north=.true. l_obc_west=.true.	-Dkey_fes2004 l_obc_tide=.true. l_tide_harmcompo=.true.	Nothing	l_obc_mars=.true.

namelist variables : 1st configuration
OBC 3D – BISC rank 1
Dirichlet – Characteristics

Type of dynamics	Where	Type of forcing	Variable	Operator
Tide dynamics	l_obc_north=.true. l_obc_west=.true. + Path file_obc_ogcm_n file_obc_ogcm_w	l_obc_tide=.true. AND l_obc_ogcm=.true.	l_obc_ogcm_rt=.true. l_obc_ogcm_rs=.true.	l_obc_diri=.true. OR l_obc_char=.true.
OGCM dynamics		l_obc_ogcm=.true.	l_obc_ogcm_rssh=.true. l_obc_ogcm_ruv=.true. l_obc_ogcm_ruvz=.true. l_obc_ogcm_rt=.true. l_obc_ogcm_rs=.true.	
Tide dynamics + OGCM dynamics		l_obc_tide=.true. AND l_obc_ogcm=.true.		

namelist variables : 1st configuration
OBC 3D – BISC rank 1
old MARS operator

Return to list of namelist variables

Type of dynamic	Where	Type of forcing	Variable	Operator
Tide dynamics	l_obc_north=.true. l_obc_west=.true. + Path file_obc_ogcm_n file_obc_ogcm_w	l_obc_tide=.true. AND l_obc_ogcm=.true.	l_obc_ogcm_rt=.true. l_obc_ogcm_rs=.true.	l_obc_mars=.true.
OGCM dynamics		l_obc_ogcm=.true.	l_obc_ogcm_rssh=.true. l_obc_ogcm_rt=.true. l_obc_ogcm_rs=.true.	
Tide dynamics + OGCM dynamics		l_obc_tide=.true. AND l_obc_ogcm=.true.		

namelist variables

Return to list of namelist variables

(M) : the variable is inside paramain.txt
(C) : the variable is inside paracom.txt
(S) : the variable is inside paraspec.txt
(E) : the variable is inside parasubs.txt

RIVERS, OUTFLOWS,TRAJECTORIES and OTHER FUNCTIONALITIES…			
namelists	**variables**	**cpp key**	**observation**
namriv (S)	file_river = '../../inputs/.dat'		Define river discharges
namtraj (S)	file_trajec='tache.dat'		Define the number and type of trajectories
namdiag (S)	l_diag=.false. date_startdiag = '01/01/1996 00:00:00' date_enddiag = '02/01/2001 18:00:01' file_diag='diag.gdg51' l_points=.false. file_point='points.dat'		
namecosys (E)	file_outflow = '../../inputs/outflow.dat' filevardiag	key_substance	Define outflows Define diagnoses variables

namelist variables

Return to list of namelist variables

TIME INTERPOLATION			
namelists	**variables**	**cpp key**	**observation**
nmlinterp (M)	perchrono=0.1 Tobs=12.0 (hours) l_champroche=.false.		Used if reading of one data.dat file do not use

(M) : the variable is inside paramain.txt
(C) : the variable is inside paracom.txt
(S) : the variable is inside paraspec.txt
(E) : the variable is inside parasubs.txt

Return to Summary

namelist variables

Return to list of namelist variables

(M) : the variable is inside paramain.txt
(C) : the variable is inside paracom.txt
(S) : the variable is inside paraspec.txt

TIDE			
namelists	**variables**	**cpp key**	**observation**
nammaree (C)	l_tide_M2harm=.false.	key_tide_fes2004 key_tide_schwid	
	l_tide_M2statcoef=.T.,tide_M2coef=0.0		effective if l_tide_M2harm=.true.
	file_tide_M2coef='bidon.dat‘		effective if l_tide_M2harm=.T. + l_tide_M2statcoef=.F.
	file_tide_harmcp='../../inputs/fes2004.nc‘ file_tide_harmcp='../../inputs/schwiderski.nc’	key_tide_fes2004 key_tide_schwid	
	l_tide_admittance=.false. datepm='02/02/2003 14:11:00' l_genpot=.false.		On progress On progress

namelist variables

ATMOSPHERICAL FIELDS			
namelists	**variables**	**cpp key**	**observation**
nammeteo (S)	imeteo_dragtype=0, cds = 0.0016		type of surface drag coefficient 0 -> constant value (= cds) if imeteo_exchtype=0 1 -> large and pond (1981) 2 -> smith and banke (1975) 3 -> geernaert et al. (1986) 4 -> charnock's relation (1955)
	imeteo_exchtype=0		dependency on air-sea temperature difference for surf drag coefficient and thermal exchange coefficient = 0 -> no dependence on tdif = 1 -> as function of tdif
	paref=101500.0 l_sflx_rpa=.false. … name_sflx_sat='sat' …		imeteo_dragtype and imeteo_exchtype not used if l_meteo_hom=T or l_meteo_stat=T

namelist variables

Return to list of namelist variables

(M) : the variable is inside paramain.txt
(C) : the variable is inside paracom.txt
(S) : the variable is inside paraspec.txt

WIND FIELD			
namelists	**variables**	**cpp key**	**observation**
nammeteoc (C)	l_meteo_stat = .true. l_meteo_hom = .true. wind_veloc = 0.0 Wind_dir = 275.0 file_meteo = 'bidon.dat'		To use constant wind field or to read a file .dat
RADIATIVE HEAT FLUX			
namelists	**variables**	**cpp key**	**observation**
namthermo (S)	l_sflx_radlossbot=.false. l_sflx_solarcst=.true. sflx_solarcst=0.0		

namelist variables

(M) : the variable is inside paramain.txt
(C) : the variable is inside paracom.txt
(S) : the variable is inside paraspec.txt

STATE EQUATION			
namelists	**variables**	**cpp key**	**observation**
nmlthermo (M)	rhoref=1027.34 saliref=35.5 tetaref=10.0 l_stateeq_lin=.true. saliref_lin=35.5 tetaref_lin=10.0		 3D 3D 3D
THERMODYNAMICS			
namelists	**variables**	**cpp key**	**observation**
nmlthermo (M)	rhoair=1.25 chp=3986.0 coext=0.1		
namthermo (S)	l_sflx_rrad=.false. l_sflx_rir=.false. …		If .false. : computation of incident SW flux If .false. : computation of infra-red flux

namelist variables

Return to list of namelist variables

(M) : the variable is inside paramain.txt
(C) : the variable is inside paracom.txt
(S) : the variable is inside paraspec.txt

EQUATION OF MOVEMENT			
namelists	**variables**	**cpp key**	**observation**
namfrot (S)	l_incbotstress=.true. botstressmax=3.0		On progress 2D
ADVECTION OF TRACERS			
namelists	**variables**	**cpp key**	**observation**
nmladvtra (M)	qmax = 8.0 qmaxz = 10.0		3D

namelist variables

(M) : the variable is inside paramain.txt
(C) : the variable is inside paracom.txt
(S) : the variable is inside paraspec.txt

VISCOSITY			
namelists	**variables**	**cpp key**	**observation**
namvisc (S)	l_smagor=.false. vismin=0.1 vismax=700.0 cosmag=0.27 fvisc=5.0 sponge_nbcell=5 vismul=5.0 l_spongs=.false. l_spongn=.false. l_sponge=.false. l_spongw=.false.		3D vismin, vismax, cosmag used for l_smagor=T
DIFFUSION			
namelists	**variables**	**cpp key**	**observation**
namdiff (S)	kx = 1.0 ky = 1.0		

namelist variables

Return to list of namelist variables

(M) : the variable is inside paramain.txt
(C) : the variable is inside paracom.txt
(S) : the variable is inside paraspec.txt

TURBULENCE

namelists	variables	cpp key	observation
namturb (S)	z0bot=0.0035		3D
	z0surf		3D
	hminfrot		2D
	hmaxfrot		2D
	turb_nbeq =0		2D
	turb_0eq_option =2		3D
	turb_2eq_option=2		3D
	bgdiff=1.0e-6		if turb_nbeq=2 3D
	nzinit=1.0e-4		3D
	kzinit=1.0e-4		3D
			3D
	l_stability=.false.		
			On progress

Parametrization of vertical mixing

$$\frac{1}{D}\frac{\partial(\frac{nz}{D}\frac{\partial u}{\partial \sigma})}{\partial \sigma} \qquad nz, kz$$

Turbulence models

2 equations : turb_nbeq=2 $\qquad nz, kz = f(k, kl, l) \quad ; \quad k \, et \, kl \, computed \; ; \quad l \, evaluated$

turb_2eq_option=1	k-kl model
turb_2eq_option=2	k-epsilon model
turb_2eq_option=3	k-omega
turb_2eq_option=4	generic length scale model

namelist variables

Return to list of namelist variables

(M) : the variable is inside paramain.txt
(C) : the variable is inside paracom.txt
(S) : the variable is inside paraspec.txt

OFFLINE To simulate trajectories or substance patches without running the dynamics (previously saved in flux form)			
namelists	**variables**	**cpp key**	**observation**
namoffline (S)	l_bz_offline = .true. path_offline = './offline_path/'		3D

namelist variables

Return to list of namelist variables

(M) : the variable is inside paramain.txt
(C) : the variable is inside paracom.txt
(S) : the variable is inside paraspec.txt

NETCDF REFERENCE (OCO 1.2)			
namelists	**variables**	**cpp key**	**observation**
nmlconvention (M)	riog_valid_min_...=-50.0 riog_valid_max_...=5000.0 cg_conv_title=CONF cg_conv_....		The min/max valid limits are used to pack the values (to save them in short type) in the netCDF output file.

namelist variables parasubs.txt (1/2)

ECOSYSTEM			
namelists	**variables**	**cpp key**	**observation**
namecosys	file_outflow	key_substance	File defining outflows
	name_filesubs		File defining substances
	filevardiag		File defining diagnoses variables
	date_startsub		Starting date of computation of ecosystem
	l_restart_subs		.true. If all substances are read in restart file
	l_obc_ogcm_rc		.true. If substance boundary condition is read from open boundary file with name after
	file_obc_subs_s (_n;_w ; _e)		
	l_out_log		Logarithm format for saving data
	l_cvrain_readfile		.true. If concentrations in rain are reading in file
	l_subflxatm_readfile		.true. If atm. fluxes are reading in file (*Rk: read one file per substances only for substances whom deposit (or conc. in rain) is given different to 0 in variable.dat)*
	sflx_sub_atm_depth		Depth until which the atmospheric flux related to substances is ditributed

Return to Summary

namelist variables parasubs.txt (2/2)

Return to list of namelist variables

ECOSYSTEM			
namelists	**variables**	**cpp key**	**observation**
namecosys	l_bilan	key_substance	.true. For computation of balance, stocks and fluxes
	name_bilfil		File name where zones and borders are defined ='' if only one budget zone which cover all the domain
	dtout_budget		Time step for budgets and fluxes output (hour)
	nb_border		Number of border (=1 if name_bilfil='')
	l_driv-cst		.true. If driving variables are constant in time (default)
	dt_driv		Time step for estimation of driving variables (used if l_driv_cst=.false.)

available cpp keys

available cpp keys (1/11)

$$\frac{\partial u}{\partial t} + L(u) - fv = -g\frac{\partial \zeta}{\partial x} - \frac{1}{\rho_0}\frac{\partial Pa}{\partial x} + \pi_x + \frac{1}{D}\frac{\partial(\frac{nz}{D}\frac{\partial u}{\partial \sigma})}{\partial \sigma} + F_x$$

Variable in namelist :
(M) : paramain.txt
(C) : paracom.txt
(S) : paraspec.txt

UNLINEAR TERMS			
Description	**Variables namelist**	**cpp key**	**Observation**
centered			default
QUICK		-Dkey_dyn_adv_quick	
QUICKEST		-Dkey_dyn_adv_quickest	
none		-Dkey_dyn_adv_linear	

INTERNAL PRESSURE GRADIENT			
Description	**Variables namelist**	**cpp key**	**Observation**
Density jacobien			default
Density jacobien + cubic spline approx		-Dkey_dyn_pg_djcs	Shchepetkin et al. 2004

available cpp keys (2/11)

$$\frac{\partial u}{\partial t} + L(u) - fv = -g\frac{\partial \zeta}{\partial x} - \frac{1}{\rho_0}\frac{\partial Pa}{\partial x} + \pi_x + \frac{1}{D}\frac{\partial(\frac{nz}{D}\frac{\partial u}{\partial \sigma})}{\partial \sigma} + F_x$$

Variable in namelist :
(M) : paramain.txt
(C) : paracom.txt
(S) : paraspec.txt

TIDE			
Description	**Variables namelist**	**cpp key**	**Observation**
Island-Portugal shelf (5km) Schwiderski (1980)	l_tide_harmcompo=.true. (S)	-Dkey_tide_schwid	default
FES2004 (MOGD)	l_tide_harmcompo=.true.	-Dkey_tide _fes2004	
file SHOM Channel Atlantic Fr	l_tide_harmcompo=.true.	-Dkey_tide _shom	On progress

TIDE - WETDRYING			
Description	**Variables namelist**	**cpp key**	**Observation**
Flux Corrected Transport to keep positive water depths in wet-drying area		-Dkey_dynh_wetdry_fct	

available cpp keys (3/11)

$$\frac{\partial u}{\partial t} + L(u) - fv = -g\frac{\partial \zeta}{\partial x} - \frac{1}{\rho_0}\frac{\partial Pa}{\partial x} + \pi_x + \frac{1}{D}\frac{\partial(\frac{nz}{D}\frac{\partial u}{\partial \sigma})}{\partial \sigma} + F_x$$

Variable in namelist :
(M) : paramain.txt
(C) : paracom.txt
(S) : paraspec.txt

OPEN BOUNDARY CONDITIONS			
Description	**Variables namelist**	**cpp key**	**Observation**
Rank : reading and saving for straight open boundary			default
Rank : reading with the use of a rotated child grid		-Dkey_grid_rotated	Longitude and latitude read from the bathy file
Rank : saving for straight open boundary but a rotated child grid	file_bathy_child='file.nc' (S)	-Dkey_tide_saveobcrotated	Define imin_child… variables in parameter.F90
Direct offline coupling			Use extract tool to prepare your obc fields from the OGCM to your MARS configuration

available cpp keys (4/11)

$$\frac{\partial u}{\partial t} + L(u) - fv = -g\frac{\partial \zeta}{\partial x} - \frac{1}{\rho_0}\frac{\partial Pa}{\partial x} + \pi_x + \frac{1}{D}\frac{\partial\left(\frac{nz}{D}\frac{\partial u}{\partial \sigma}\right)}{\partial \sigma} + F_x$$

Variable in namelist :
(M) : paramain.txt
(C) : paracom.txt
(S) : paraspec.txt

VERTICAL DIFFUSION (nz,kz)			
Description	**Variables namelist**	**cpp key**	**Observation**
constant	turb_nbeq=0, turb_0eq_option=1 (S)		
Prandt	turb_nbeq=0, turb_0eq_option=2 (S)		
Quetin	turb_nbeq=0, turb_0eq_option=3 (S)		
Pacanovski et Philander, 1981	turb_nbeq=0, turb_0eq_option=4 (S)		
Gaspard et al., 1990	turb_nbeq=1 (S)		
K-kl	turb_nbeq=2, turb_2eq_option=1 (S)		
K-epsilon	turb_nbeq=2, turb_2eq_option=2 (S)		
K-omega	turb_nbeq=2, turb_2eq_option=3 (S)		
Generic 2 eq model	turb_nbeq=2, turb_2eq_option=4 (S)		

Variable in namelist :
(M) : paramain.txt
(C) : paracom.txt
(S) : paraspec.txt

available cpp keys (5/11)

$$\frac{\partial u}{\partial t} + L(u) - fv = -g\frac{\partial \zeta}{\partial x} - \frac{1}{\rho_0}\frac{\partial Pa}{\partial x} + \pi_x + \frac{1}{D}\frac{\partial}{\partial \sigma}\left(\frac{nz}{D}\frac{\partial u}{\partial \sigma}\right) + F_x$$

VISCOSITY			
Description	**Variables namelist**	**cpp key**	**Observation**
constant	L_smagor=.false. (S)		
Smagorinsky, 1963	L_smagor=.true. (S)		
	Sponge layer at each open boundary		

OPEN BOUNDARY CONDITIONS			
Description	**Variables namelist**	**cpp key**	**Observation**
Rank : reading and saving for straight open boundary			default
Rank : reading with the use of a rotated child grid		-Dkey_grid_rotated	Longitude and latitude read from the bathy file
Rank : saving for straight open boundary but a rotated child grid		-Dkey_tide_saveobcrotated	Define imin_child… in parameter.F90
Direct offline coupling			

available cpp keys (6/11)

$$\frac{1}{D}\frac{\partial p}{\partial \sigma} = -\rho g$$

Variable in namelist :
(M) : paramain.txt
(C) : paracom.txt
(S) : paraspec.txt

STATE EQUATION			
Description	**Variables namelist**	**cpp key**	**Observation**
linear	l_stateeq_lin=.true. (M)		
	l_stateeq_lin=.false. (M)		

VERTICAL DISCRETIZATION			
Description	**Variables namelist**	**cpp key**	**Observation**
sigma	l_equisig=.true. Or specify sig for each level (S)		
generalized sigma	l_equisig=.true. (S) hc = 5.0 theta_sig=0.000001 b_sig=1	-Dkey_siggen	Do not change let theta_sig go to zero to get pure sigma coordinates

available cpp keys (7/11)

$$\frac{\partial DT}{\partial t}+\frac{\partial D(uT-k_x\frac{\partial T}{\partial x})}{\partial x}+\frac{\partial D(vT-k_y\frac{\partial T}{\partial y})}{\partial y}+\frac{\partial D(w^{å}T-\frac{kz}{D^2}\frac{\partial T}{\partial \sigma})}{\partial \sigma}=\frac{1}{\rho_0 C_p}\frac{\partial I}{\partial \sigma}$$

ADVECTION OF TRACERS		
Description	**cpp key**	**Observation**
Along x and y		
QUICK/upstream		default
QUICKEST scheme + ULTIMATE limiter	-Dkey_tssub_adv_ultimatequickest	
Q+U+MACHO splitting	Dkey_tssub_adv_ultimatequickestmacho	
Along z		
Centered		default
Quick/upwind	-Dkey_tssub_wquickupwind	
Upwind Compact and conservative	-Dkey_tssub_wcompact	

available cpp keys (7/11)

$$\frac{\partial DT}{\partial t}+\frac{\partial D(uT-k_x\frac{\partial T}{\partial x})}{\partial x}+\frac{\partial D(vT-k_y\frac{\partial T}{\partial y})}{\partial y}+\frac{\partial D(w^{\mathring{a}}T-\frac{kz}{D^2}\frac{\partial T}{\partial \sigma})}{\partial \sigma}=\frac{1}{\rho_0 C_p}\frac{\partial I}{\partial \sigma}$$

DIFFUSION OF TRACERS			
Description	**Variables namelist**	**cpp key**	**Observation**
along sigma			default
along geopotentials		-Dkey_ts_diffh_geo	

available cpp keys (8/11)

Variable in namelist :
(M) : paramain.txt
(C) : paracom.txt
(S) : paraspec.txt

$$\frac{\partial DT}{\partial t}+\frac{\partial D(uT-k_x\frac{\partial T}{\partial x})}{\partial x}+\frac{\partial D(vT-k_y\frac{\partial T}{\partial y})}{\partial y}+\frac{\partial D(w^{å}T-\frac{kz}{D^2}\frac{\partial T}{\partial \sigma})}{\partial \sigma}=\frac{1}{\rho_0 C_p}\frac{\partial I}{\partial \sigma}$$

HEAT FLUXES			
Description	**Varia namelist**	**cpp key**	**Observation**
Solar flux Bulk MAST 3 (Luyten et al, 1992) Bulk NOMADS2 (Gill, 1982) Bulk NOMADS2 (Gill, 1982) Read	l_sflx_rsolar=.true. (S)	-Dkey_sflx_solar_luyten -Dkey_sflx_solar_gill	
Thermal flux Swimbank in Agoumi Phd Luyten and De Mulder (1992) Berliand and Berliand (1952) Read	l_sflx_rir=.true. (S)	-Dkey_sflx_ir_swimbank -Dkey_sflx_ir_luyten -Dkey_sflx_ir_berliand	
Turbulent fluxes Clark et al (1995), Elliot and Clark (1990) Luyten and De Mulder (1992) Large et Yeager (2004) Fairall (2003)	imeteo_exchtype=1 (S)	-Dkey_sflx_turb_default -Dkey_sflx_turb_luyten -Dkey_sflx_turb_large -Dkey_sflx_turb_fairall	Coef depends on diff $t_{air-mer}$

available cpp keys (9/11)

Variable in namelist :
(M) : paramain.txt
(C) : paracom.txt
(S) : paraspec.txt

OPEN BOUNDARY CONDITIONS			
Description	**Varia namelist**	**cpp key**	**Observation**
Nesting with an OGCM Nesting with a previous rank or from harmonics	l_obc_ogcm (S) l_obc_tide (S)		
Where are the open boundaries **Which** obc file	l_obc_north…(S) file_obc_ogcm_n…		
MARS standard Sea surface height is prescribed Gradient nul for velocity	l_obc_mars (S)		Initial and default operator
CYCLIC Zonaly or meridionaly	l_obc_cycl (S) l_obc_cycl_x l_obc_cycl_y		Choose another operator for open boundaries that are not cyclic
CARACTERISTICS METHOD	l_obc_char (S)		2D part only MARS standard for 3D variables
DIRICHLET SSH prescribed (strong condition) Velocities prescribed (weak condition)	l_obc_diri (S)		

available cpp keys (10/11)

PASSIVE SUBSTANCES		
-Dkey_substance		Sequential, openMP,MPI
BIOLOGY		
-Dkey_substance -Dkey_biolo		Sequential, openMP,MPI
SEDIMENTOLOGY		
-Dkey_substance -Dkey_sedim		Sequential, MPI
CHEMISTRY OF CONTAMINANTS		
-Dkey_substance -Dkey_contaminant		Sequential, MPI
Rq : input files and modules of sedimentology, biology and chemistry of contaminants are distincts		
ALL MODULES		
-Dkey_subs_part_eqsubdt	Substances with settling velocity are advected with sub time scheme. This key allows an optimization of the calculation	

available cpp keys (11/11)

Variable in namelist :
(M) : paramain.txt
(C) : paracom.txt
(S) : paraspec.txt

AGRIF PACKAGE		
-Dkey_agrif -Dkey_agrif_use -Dkey_agrif -Dkey_agrif_use –DAGRIF_MPI –Dkey_MPI_2D	Compile with –r8	Sequential MPI
TRAJECTOIRES		
-Dkey_trajec2d -Dkey_trajec3d	2D Simulation (*not maintained*) 3D Simulation	Sequential, openMP Sequential, openMP
MPI		
-Dkey_MPI_2D	Several output files if l_out_nc4par= .false. (S) One single output file if l_out_nc4par= .true (S)	
MPI + OMP		
-Dkey_MPI_2D -Dkey_MPIOMP	Several output files if l_out_nc4par= .false. (S) One single output file if l_out_nc4par= .true (S)	

Input files *.dat

under $RDIR/CONF/inputs

- ✓ Setting up of output files : output.dat
- ✓ Setting up of tracking points : points.dat
- ✓ Setting up of river discharges : river.dat
- ✓ Setting up of outflow : outflow.dat
- ✓ Setting up of substances (key_substance) : variable.dat
- ✓ Setting up of budgets domains and borders : bilmat.dat

Examples of these inputs files are also in $HOMEMARS/../EXAMPLES/

Parametrization of the configuration : file output.dat

setting up of output files

The name of the file output.dat is set in paraspec.dat (file_output)

THREE REQUIREMENTS :

1. THE MAIN OUPUT MUST BE DEFINED AT FIRST
2. The user can choose l_demer=.true. or l_out_precdate=.true. (both at the same time is not allowed)
3. The choice l_demer=.true. or l_out_precdate=.true. is possible just once inside the file output.dat.

```
************************************************************************
Champs                  ,| the name of the output file will be champs_suffix.nc, suffix being defining in paracom.txt
06/03/2005 00:00:00     ,| date_startout
09/03/2005 00:00:00     ,| date_endout
1.0d0                   ,| pasor (real in double precision)
.true.                  ,| l_posit
0 421 0 501             ,| iextmin,iextmax,jextmin,jextmax
1 30                    ,| kextmin,kextmax  (between 1 and kmax)
1 1 1                   ,| i-step,j-step,k-step
.false.                 ,| l_filebydate
.false.                 ,| l_out_off
.false.                 ,| l_out_precdate
.false.                 ,| l_demer
.true.                  ,| l_out_ssh
.true.                  ,| l_out_ubt
.false.                 ,| l_out_u3d
.true.                  ,| l_out_sal
.false.                 ,| l_out_temp
.false.                 ,| l_out_wind
.false.                 ,| l_out_visc
.false.                 ,| l_out_turb
.false.                 ,| l_out_atmsflx
************************************************************************
```

Points.dat
Setting up of tracking points

The name of the file points.dat is set in paraspec.dat (file_point)

- **********
- Point1 ,| prefix of output file name
- 06/03/1900 00:00:00 ,| starting date
- 06/03/2500 00:00:00 ,| ending date
- 600.0 ,| time step (s)
- .true. ,| location in i,j,k (.true.) or in lon,lat,k (.false.)
- 29 29 12 ,| location
- .true. ,| l_out_ssh_track
- .false. ,| l_out_ubt_track
- .true. ,| l_out_u3d_track
- .true. ,| l_out_sal_track
- .true. ,| l_out_temp_track
- **********

river.dat
Setting up of river discharges

The name of the file river.dat is set in paraspec.dat (file_river)

- **********
- **Seine** ,| river name
- **.true.** ,| location in i,j,k (.true.) or in lon,lat,k (.false.)
- **142 45 12** ,| location
- **100** ,| thickness (m) on which the outflow is distributed
- **.false.** ,| .true. for a stationary river discharge
- **450.0** ,| value of the stationary river discharge
- **../../inputs/debits/deb-seine-nc.dat** ,| river discharges data
- **.true.** ,| set to .true. for reading the river temperature
- **../../inputs/concentrations/temp-seine.dat** ,| data of temp
- **10.0 6.0** ,| mean temp of river (degrees) & annual amplitude (degrees)
- **********

River thickness : estimated downwards from the interface of the layer k
- If the river thickness is set to 0., the outflow is completed at the layer k
- If the river thickness is higher than depth, the outflow is distributed over the total depth
- To prescribe the river discharge at the bottom, set k=1 and river_thickness=0.

outflow.dat
Setting up of outflows

The name of the file outflow.dat is set in parasubs.dat (file_outflow)

```
************************************************************
seine              , ! name of outflow (= river name)
.true.             ,! location of outflow : i,j,k (.true.)/lon,lat,k (.false.)
142 45 12          , ! location_i location_j location_k
.false.            , ! flux data (.true.) or concentration data
seine              , ! name of the river relative to the outflow
                     (river discharge used jointly with tracer concentration to estimate the flux ;
                       not used if previous parameter is .T.)
1                  , ! number of advected tracer (unused)
nitrate            , ! name of rejected tracer
../../inputs/concentrations/N-NO3-seine.dat
                    ,! name of file describing the time evolution of the outflow
2                   , ! number of advected tracer (unused)
silicium_dissous   ,! name of rejected tracer
************************************************************
```

variable.dat (1/4)
Setting up of variables

The name of the file variable.dat is set in parasubs.dat (name_filesubs)

Begining of the file :

➢**********

1 ,| number of variable (unused)
nitrate ,| name of variable (unique word)
nitrate_NO3 ,| LONG_NAME of variable (NetCDF OCO format for gridded data products)
mole_concentration_of_nitrate_in_sea_water ,| STANDARD_NAME of variable (NetCDF OCO format for gridded data products)
micromoleN.l-1 ,| unit of concentration of variable
-0 ,| minimum valid value of variable (NetCDF OCO format for gridded data products
+100 ,| maximum valid value of variable (NetCDF OCO format for gridded data products

……..following the file next few slides

variable.dat (2/4)
Setting up of variables (dissolved and fixed)

```
.....
+100    ,| maximum valid value of variable (NetCDF OCO format for gridded data products
DISS            ,| type of variable
                                    [GRAV,SAND,MUDS,PART,SORB/NCSP,DISS,FIXE,DRIV,INTE]
.true.          ,| pelagic variable (unused)
.true.          ,| benthic variable (unused)
 == if particulate variable : add 4 more lines below
 == if state variable      : add 5 more lines below
0.000           ,| t90 (time after which 90% of rejected matter has disappeared) in hours
0.000           ,| uniform atmospheric deposition (unit/m2/s) (not used for FIXE)
0.000           ,| concentration in rainwater (unit/m3 of water)
5.00000         ,| initial concentration in water column (unit/m3)
10.00000        ,| initial concentration in sediment (in % if SAND, GRAV, MUDS, PART; /kg if
SORB/NCSP; /m3 EI if DISS)
.true.          ,| saving in output file
none            ,| name of substance read from initial conditions file
none            ,| name of substance read from boundary conditions file
```

➢**********

variable.dat (3/4)
Setting up of particulate variables

.....
+100 ,| maximum valid value of variable (NetCDF OCO format for gridded data products
MUDS ,| type of variable [GRAV,SAND,MUDS,PART,SORB/NCSP,DISS,FIXE,DRIV,INTE]
.true. ,| pelagic variable (unused)
.true. ,| benthic variable (unused)
== if particulate variable : add 4 more lines below
0.00001,0.0005,10,0.085,0.5,60,0,| 8 parameters of settling velocity
[only the **second parameter** is used if not key_sedim, = **constant settling velocity**
[see module Sediment if key_sedim)
10. ,| critical stress of deposition (used only if key_sedim)
2600. ,| density of particle (used only if key_sedim)
0.00002 ,| diameter of particle (used only if key_sedim)
== if state variable : add 5 more lines below
..... *same as for dissolved variables (see previous slide)*

➢**********

variable.dat (4/4)
Setting up of variables (driving, intermediate)

```
.....
DRIV            ,| type of variable [GRAV,SAND,MUDS,PART,SORB/NCSP,DISS,FIXE,DRIV,INTE]
.true.          ,| pelagic variable (unused)
.true.          ,| benthic variable (unused)
 == if particulate variable : add 4 more lines below
 == if state variable       : add 5 more lines below
.true.          ,| saving in output file
none            ,| name of substance read from initial conditions file
none            ,| name of substance read from boundary conditions file

➢**********************************************************
```

bilmat.dat
Setting up of variables

The name of the file bilmat.dat is set in parasubs.dat (name_bilfil)

See budget functionnality

➢**********

1 ,| number of the border
Zone_1 ,| name of border or budget domain
.true. ,| .true. if border is close to build a domain
4 ,| total number of horizontal (W-E) and vertical (N-S) segments
100,200,20,S , | i1,i2, j1, South
200,20,50,E , | i2, j1,j2, East
200,100,50,N , | i2,i1, j2, North
100,50,20,W , | i1, j2,j1, West

➢**********

vardiag.dat
Setting up of diagnostic variables

The name of the file vardiag.dat is set in parapec.dat (file_diag)
(operationnel only with key_biolo)

➢**********

```
1                                   ,| number of the diagnostic variable
maximum_de_diat          ,| name of diagnostic variable (unique word)
maximum_de_diat          ,| LONG_NAME of variable
maximum_de_diatom_mass_concentration_in_sea_water    ,| STANDARD_NAME
micromole.l-1    ,| unit of concentration of variable
-0          ,| minimum valid value of variable (NetCDF OCO format for gridded data products
+100     ,| maximum valid value of variable (NetCDF OCO format for gridded data products
3           ,| dimension  (1=1D;  2=2D  ; 3=3D)
.true.     ,| Variable in water (k,i,j)
.false.    ,| Variable in sediment (i,j,k)
.true.     ,| saving in file
```

➢**********

Module “trajectory”

How to simulate trajectories ? (1/2)

$UDIR/CONF/CONF-CASE directory

- **Makefile.caparmor[_rankX]**
  - add -Dkey_trajec2d (*not maintained*) OR -Dkey_trajec3d in CPPFLAGS
- **gmake clean ; gmake**

Rq : only for simple particle tracking.

if you want to simulate biological behaviour and other biological features (growth...) then use -Dkey_ibm without -Dkey_traject3d

$RDIR/CONF/CONF-CASE directory

- Define the associated variables in namelist paraspec.txt
  - (*see details next slide*)
- Set up the file file_trajec in inputs directory (*see next slide*)

How to simulate trajectories ? (2/2)

Configure trajectories in file paraspec.txt

****

- file_trajec='../../inputs/traject3d_type3ncdf.dat' : name of the file where particle patches are defined. Different structure of the file depending on circle, rectangular or netcdf defined patch
- itypetraj = 3 : 1 (circle patch), 2 (rectangular patch) or 3 netcdf patch)
- ndtz = 50 : number of discretisation within dt for random walk, i.e. Time step of RW will be dt/ndtz. It has to be no more than a few seconds.

****

Fill the file_trajec file in input directory with as many patch as necessary, following provided examples. ! Different structure of file depending on itypetraj !

For itypetraj=3, youn need to give the name of a netcdf file which has the following structure (same as netcdf structure of the output file of trajectory module :

dimensions:

traj = 50200 ;

time = 1 ; not considered yet, so time is taken from definition in file_trajec.

variables:

double time(time) ;

double longitude(time, traj) ;

double latitude(time, traj) ;

float DEPTH(time, traj) ;

Module “IBM” (Individual Based Model)

How to use IBM ? (1/2)

$UDIR/CONF/CONF-CASE directory

- Makefile.caparmor[_rankX]
  add -Dkey_ibm
- gmake clean ; gmake

Rq : do not add -Dkey_trajec3d. Trajectory module will be called automatically from module ibm

$RDIR/CONF/CONF-CASE directory

- Define parameters in namelist paraibm.txt
- Set up the file file_trajec in inputs directory

How to use IBM ? (2/2)

Adapt existing case studies to your case study

- Create your own key-species and add in Makefile-caparmor[_rankX]

Simplified structure of the module :

```
STEP
    CALL IBM_INIT
        Read namelist paraibm.txt and initialise several particle's properties
        CALL TRAJ_INITSAVE ; common with traj module to initiate patch 3D position and dates
    CALL IBM_3d
        IF larva (pelagic)
            CALL TRAJ ; common with trajectory module
        ENDIF
        IF adult mobile
            CALL MOVEMENT ; not ready
        ENDIF
      Calculate egg development, larval growth, vertical velocity, etc...
        CALL GROWTH ; no generic module available
        CALL MORTALITY ; simple (reduction of number of super-individual with constant mortality)
```

Available generic subroutines in the module that can be used :

- Buoyancy, vertical migration, detection of pycnocline

Module “AGRIF” Adaptative Grid Refinement In Fortran

How to simulate use Agrif ?

$UDIR/CONF/CONF-CASE directory

- **Makefile.caparmor[_rankX]**
  - add -Dkey_agrif -Dkey_agrif_use in CPPFLAGS
- **Gmake clean ; gmake**

$RDIR/CONF/CONF-CASE directory

- **Prepare a paraspec.txt for each grid. Pay attention to the file units of output files (iscreen, iscreenlog, ierrorlog, iwarnlog) to avoid conflicts.**
- **If substances, prepare a parasubs.txt (parabiolo.txt, parasedim.txt) for each grid.**
- **In the namelist files (para*.txt) of the agrif zoom, all the *.dat files must be different from the mother's ones except variable.dat and vardiag.dat.**
- **Define a grid hierarchy from AGRIF_fixed_Grids.in**

AGRIF : Exemples of AGRIF_fixed_Grids.in files

AGRIF_fixed_Grids.in file (1 grid)	explanation
1 164 176 93 102 5 5 5 0	Number of AGRIF embbeded grids ileft,iright,jbottom,jtop (location in the mother grid), x,y,time refinement factor for child grid End (no more child grid)
AGRIF_fixed_Grids.in file (2 grids coupled to same mother grid)	**explanation**
2 164 176 93 102 5 5 5 104 102 70 76 5 5 5 0 0	2 ranks Location in the mother grid and refinement Location in the mother grid and refinement Closure of grid 1 Closure of grid 2
AGRIF_fixed_Grids.in file (2 embedded grids)	**explanation**
1 104 176 53 102 5 5 5 1 10 50 20 40 3 3 3 0 0	First child grid Location in the mother grid and refinement Second child grid Location in the first grid and refinement Closure of grid 1 Closure of grid 2

Substances

Substances summary

Advected substances in MARS may have different behavior :

- dissoved passive tracers (conservative)
- particulate passive tracers (conservative with settling velocity)
- non conservative
  - first order degradation : T90
  - Biological variables (using module of BIOLOGY)
  - Contaminant variables (using module MET&OR)

✓ Definition of available types of substances
✓ Summary of how to implement MARS with substances
✓ Initial conditions
✓ Boundaries conditions
✓ Budget and fluxes computing
✓ Tracking points
✓ Diagnostic variables

Types of variables (1/2)
Definition of dissolved, particulate variables fixed, driving or intermediate variables

- Dissolved : type DISS
- Particulate
  - either type GRAV (gravel) (use key_sedim..)
  - either type SAND (use key_sedim..)
  - either type MUDS (use key_sedim..)
  - either type PART
  - either SORB (or NSCP)

GRAV, SAND, MUDS, and PART are « constitutive » variables expressed in kg/m^3 if you use key_sedim

SORB is « non constitutive » variable, expressed in mass/m^3, with the mass unit choose by user

- Fixed : type FIX : not advected but submitted to biogeochemical reactions

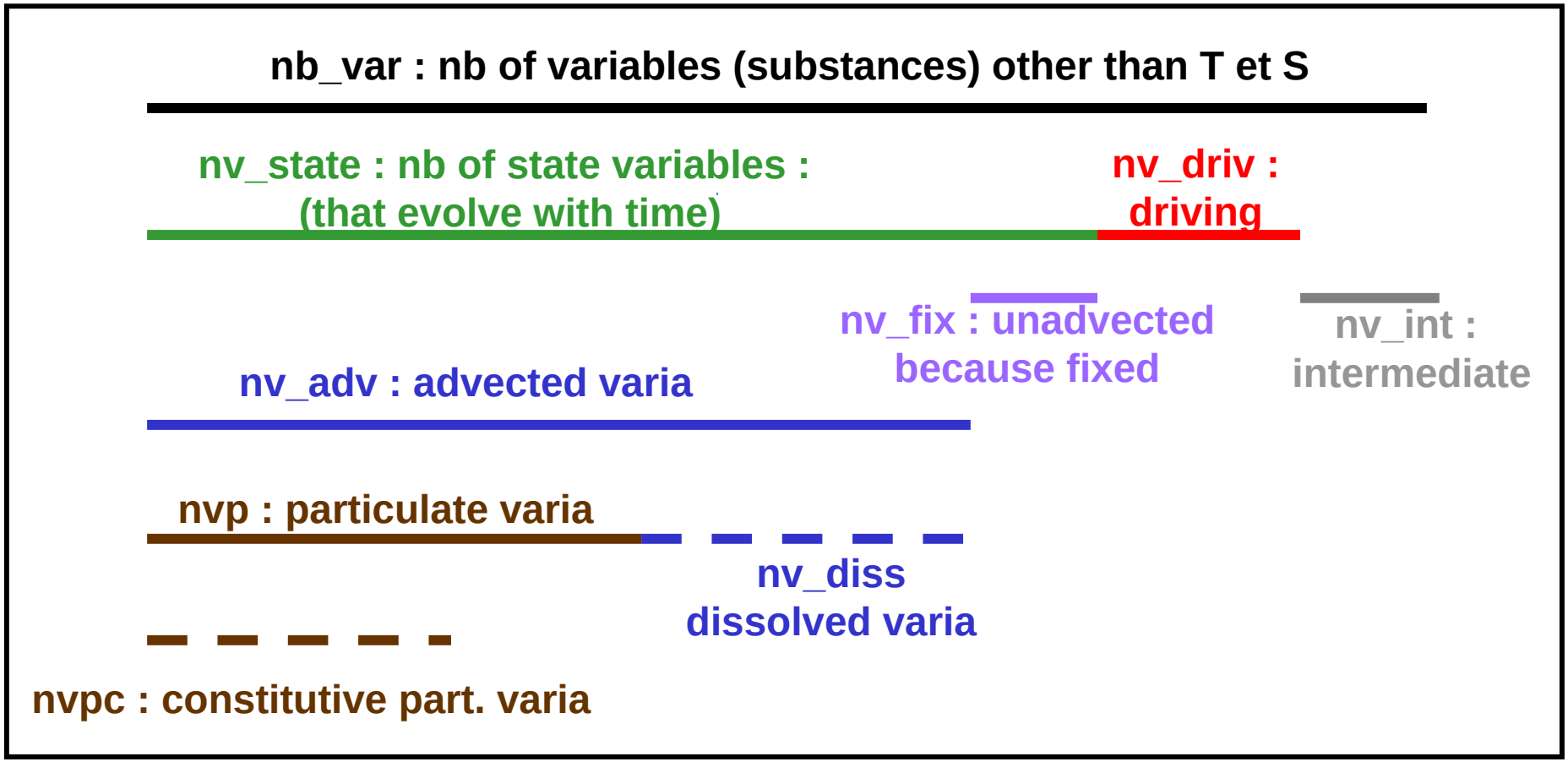
- Driving : type DRIV : not advected but defined (read from a file) in a subroutine writed by the user in *usersubstance.F90*

- Intermediate : type INTE : not advected but computed (combination of other variables) in a subroutine writed by the user in *usersubstance.F90*

Type of each simulated variable is set in variable.dat by user

Types of variables (2/2)
Definition of number and order (rank) of variables

MARS places the variables in a specific order : starting with the particulate variables (GRAV, SAND, MUD, PART, SORB), then dissolved, then fixed, then intermediate variables.



Advection of tracers (1/3)

directory $UDIR/CONF/CONF-CASE

- **Makefile.caparmor[_rankX]**
  - add -Dkey_substance dans CPPFLAGS
- **parameters.F90[_rank*]**
  - Set nb_var : total number of substances
- **gmake clean ; gmake**

Rq : the cpp key **-Dkey_substance will select green routines** inside the slide decriving the structure of the code

directory $RDIR/CONF/CONF-CASE

- **Set the variables in parasubs.txt**
  - file_outflow = '../inputs/bidon.dat'
  - name_filesubs = '../inputs/variable_cyl.dat'
  - date_startsub = '01/01/1900 00:00:00'
  - tdebsubs=datosec(datedeqsubs)
  - l_out_log=.false.
- **Set up under inputs the following files :**
  - variable.dat
  - outflow.dat

If no outflow : initialize ***file_outflow*** with a non existing file

Advection of passive tracers (2/3)

Initial conditions

- If **l_initfromfile** = .false. **(in** *paraspec.txt***)** : Initial concentrations of substances are uniform throughout the domain and equal to the value read in *variable.dat* file [*initial concentration in water column (unit/m3)]*

- If **l_initfromfile** = .true. ***(*****in** *paraspec.txt****)*** **and l_restart_sub**=.true. (**in** *parasubs.txt*) : Initial concentrations of all substances are read in the file ***file_init*** defined in ***paraspec.txt***

- If **l_initfromfile** = .true. ***(*****in** *paraspec.txt****)*** **and l_restart_sub**=.false. (**in** *parasubs.txt*) : Initial concentrations of substances are read in the file ***file_init*** only for substances with the same name (« *name of substance read from initial conditions file* » in *variable.dat*) as in the ***file_init***. For the others (name=*none* in ***variable.dat***), initial concentration is uniform and equal to the value read in *variable.dat* file [*initial concentration in water column (unit/m3)]*

- If required, modify inittssub.F90 to initialize substances

Advection of passive tracers (3/3)

Boundaries conditions

➢If **l_obc_ogcm_rc** = .false. **(in** *parasubs.txt***)** : concentrations of substances at boundaries are constant and equal to the initial value read in *variable.dat* file [*initial concentration in water column (unit/m3)]*

➢If **l_obc_ogcm_rc** = .true. ***(*in** *parasubs.txt****)*** : concentrations of substances at boundaries are read in files ***file_obc_subs_n (_s,_e,_w)*** only for substances with the same name (« *name of substance read from obc file* » in *variable.dat*) as in the ***file_obc_subs***. For the others (name=*none* in ***variable.dat***), obc concentration is constant and equal to the initial value read in *variable.dat* file [*initial concentration in water column (unit/m3)]*

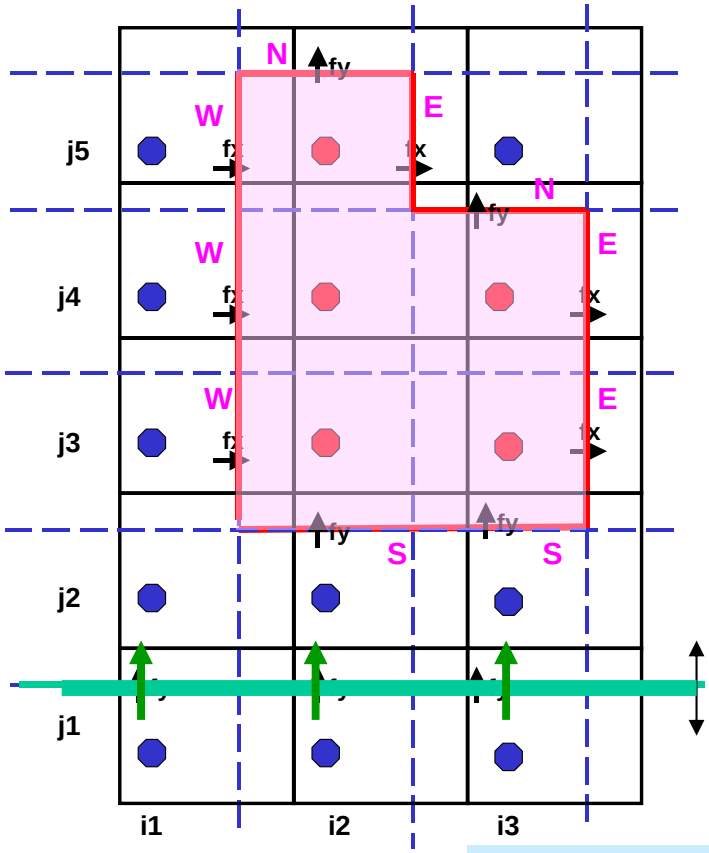
Computing Massbalance (Stock, fluxes and budgets) (1/3)

l_bilan=.true. ***in*** *parasubs.txt*

For each substance :

calculation of fluxes through boundaries defined by the user

calculation of budgets (stocks and fluxes) in sub-domains defined by the user



- **NET CUMULATED FLUXES (since the beginning of the simulation) in and out of a given sub-domain**
- **STOCKS in the water column and in the sediment within the sub- domain**
- **NET CUMULATED FLUXES in the sub-domain (since the beginning of the simulation) due to rivers inputs and exchanges with the atmosphere.**
- + If key_contaminant : cumulated fluxes by exchange between substances due to biogeochemical reactions
- **+ cumulative fluxes through open boundaries**

boundaries and sub-domains are defined from the surface water to the bottom ; a budget sub-domain is only valid if the boundary is closed

Computing Massbalance (Stock, fluxes and budgets) (2/3)

to build file : name_bilfil

subbudgetdomain.dat **: example of file defining sub-domains and boundaries (you will find it in $HOMEMARS/../EXAMPLES/)**

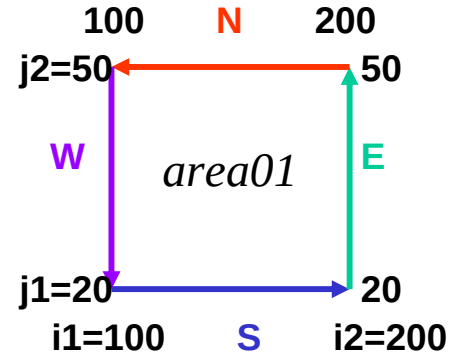
Boundaries are defined by several segments S-N and/or W-E (any number of segments)
Budgets are computed only inside sub-domains surrounded by closed boundaries

Each boundary segment is characterized by a letter N, S, W or E to determine whether this border is a north, south, west or east segment of the sub-domain

ASSIGN closed boundaries FIRST and open boundaries next

Example :

```
2                 , boundary number (or sub-domain number if closed boundary)
area01            , boundary name ( or sub-domain name)
.true.            , l_border_close
4                 , total number of horizontal (W-E)  or vertical (N-S) segments
100,200,20,S      ,  i1,i2,j1, South
200,20,50,E       ,  i2,j1,j2, East
200,100,50,N      ,  i2,i1,j2, North
100,50,20,W       ,  i1,j2,j1, West
```

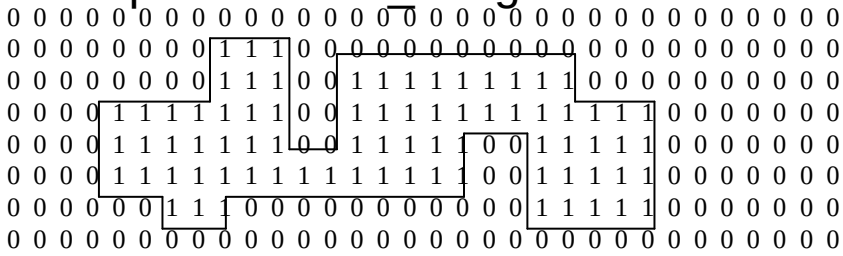


Computing Massbalance (Stock, fluxes and budgets) (3/3)

Verification of domains

in ***fic_verif_zone_budget*** created at the beginning of the simulation

Example for mask_budget areas :



Budget areas : mask_bud=1 if inside
mask_bud=0 outside

For Fluxes : mask_bord_W= 1 if cell W ; = -1 (E), 0 (none)
mask_bord_N= 1 if cell S ; = -1 (N)

Massbalance_output Files : File **.csv** (separator = ';')

1 file for each budget sub-domain (closed boundary) and 1 file for 10 variables
1 file for all open boundaries (fluxes through each boundary)

One line /time_out_budget
Massbalance_output_filenames : automatically created

1 for the first 10 variables
2 for variables 10 to 20..

outputfilename_suffix_bil[boundary number]_1.csv
outputfilename_suffix_border.csv

Obc Cumulated fluxes :

$$Flx_{cum} = \sum_t \left(Flx_W - Flx_E + Fly_S - Fly_N \right)$$ input if the slope (/time) is positive

Input cumulative fluxes :

$$Flin_{cum} = \sum_t \left(Flrej + Flatm \right)$$

Budget :

$$Bud = Stock_{eau} + Stock_{sed} - Flx_{cum} - Flin_{cum}$$

Tracking points

Useful to save vertical profiles in only a few points and with a shorter time step than in output file

- **Define the following variables inside paraspec.txt**
  - l_points=.true.
  - file_point='../inputs/points.dat'
- **Define (under inputs) the file :**
  - points.dat

Diagnostic variables

Computed by the user in the subroutine diagnostic.F90
or elsewhere, in order to save some diagnostics or indicators

(operationnel now only with key_biolo)

- **Define the following variables inside paraspec.txt**
  - l_diag=.true.
  - date_startdiag = '01/01/1900 00:00:00'
  - date_enddiag = '02/01/2001 18:00:01'
  - file_diag='diag.out'
- **Define (under inputs) the file :**
  - vardiag.dat

Module of sedimentology

&

Module of contaminant

separate manuals will be available later

contact Benedicte.Thouvenin@ifremer.fr if needed

Module of biology

separate manual will be available later

general guidelines are presented on the following slides

Simulation of biologic behaviors (1/6)

directory $UDIR/CONF/CONF-CASE

- **Makefile.caparmor[_rankX]**
  - add -Dkey_substance -Dkey_biolo dans CPPFLAGS
- **parameters.F90[_rank*]**
  - Set nb_var
  - Modify nparam_biolo=110 if required
- **gmake clean ; gmake**

Rq : the cpp key **-Dkey_substance will select green routines** inside the slide decriving the structure of the code, the cpp key **-Dkey_biolo will select red routines**.

directory $RDIR/CONF/CONF-CASE

- **Set up parabiolo.txt**
- **set (under inputs) the files :**
  - **variable.dat**
  - **vardiag.dat**
  - **outflow.dat**

List of routines relative to the module « biology » (2/6)

MODULE	ROUTINES	DESCRIPTION
	subreaddat.F90	Read name_filesubs = variables.dat
	outflow.F90	Read file_outflow = outflow.dat
varreaddat.F90	var_read_diag biolo_read_param	Read filevardiag = vardiag.dat Read file *.pbd (biolo parameters)
bioloinit.F90	biolo_userinit = biolo_init_dinodiatzoo	Special user initialisation
biolodynzwat.F90	biolo_dyn_zwat	Compuation of inter-cell effets (non-hydrauliques)

List of routines relative to the module « biology » (3/6)

MODULE	ROUTINES	DESCRIPTION
biolosinksource.F90	biolo_sksc_wat biolo_sksc_sed	Estimate dcdt, cv_wat(nv_adv+1:nv_state) Estimate cv_sed (dc)
biolodyncellwat	biolo_dyn_cellwat = incellwat_dinodiatzoo	Transformations inside a wet cell Called by biolo_sksc_wat (dc)
biolodyncellsed	biolo_dyn_cellsed = incellsed_dinodiatzoo	Transformations inside a sediment cell Called by biolo_sksc_sed (dc)

Routines to be modified by the user (4/6)

- **biolodyncellsed.F90:**
  - SUBROUTINE incellsed_dinodiatzoo(km,im,jm,c,dc)
- **biolodyncellwat.F90:**
  - SUBROUTINE incellwat_dinodiatzoo(km,im,jm,xe,c,dc)
- **biolodynzwat.F90:**
  - SUBROUTINE verti_dinodiatzoo(xe,cv_wat,sal,temp)
- **bioloinit.F90:**
  - SUBROUTINE biolo_init_dinodiatzoo(cv_wat)
- **biolosinksource.F90:**
  - SUBROUTINE biolo_sksc_sed
  - SUBROUTINE biolo_sksc_wat(cv_wat,sal,temp,xe)

Remark : if the users does not want to use dinodiatzoo subroutines, he can implement his own code inside the module and then precise which subroutine should be used via the interface

Example of interface (5/6)

MODULE biolodynzwat

- **Estimates of non-hydraulics inter-cells influences**
- **Interface**

```
  USE *
  IMPLICIT NONE
  PRIVATE

  !! * Accessibility
  PUBLIC biolo_dyn_zwat          ! routine called by step3dxy.F90 and step3dyx.F90

  !! * Interface
  INTERFACE biolo_dyn_zwat
    MODULE PROCEDURE verti_dinodiatzoo
  END INTERFACE

!! * Private variables

 CONTAINS
  !!=====================================================================

  SUBROUTINE verti_dinodiatzoo(xe,cv_wat,sal,temp)
```

MODULE varreaddat : Shared module variables (6/6)

! **number of diagnostic variables**

```
INTEGER, PUBLIC              :: ndiag_1d,     &  ! 1 dimension (time)
                                ndiag_2d,     &  ! 2 dimensions (i,j)
                                ndiag_3d,     &  ! 3 d
                                ndiag_3d_wat, &  ! 3 d (k,i,j)
                                ndiag_3d_sed, &  ! 3 d (i,j,k)
                                ndiag_tot
```

! **arrays of diagnostic variables**

```
REAL(KIND=rsh), ALLOCATABLE, DIMENSION(:), PUBLIC   :: diag_1d  !(index of var, time)
REAL(KIND=rsh), ALLOCATABLE, DIMENSION(:,:,:), PUBLIC :: diag_2d  ! (index of var, i, j)
REAL(KIND=rsh),ALLOCATABLE,DIMENSION(:,:,:,:),PUBLIC  :: diag_3d_wat !(index of var,k,i,j)
#ifdef key_sedim
REAL(KIND=rsh),ALLOCATABLE,DIMENSION(:,:,:,:),PUBLIC  :: diag_3d_sed !(index of var,i,j,k)
#endif
```

! **names of diagnostic variables**

```
CHARACTER(LEN=lchain), ALLOCATABLE, DIMENSION(:), PUBLIC  ::  name_vardiag

                                ! units of diagnostic variables
CHARACTER(LEN=lchain), ALLOCATABLE, DIMENSION(:), PUBLIC  ::  unit_vardiag

                                ! rank of diagnostic variables
INTEGER,ALLOCATABLE, DIMENSION(:), PUBLIC  :: irk_diag    ! (reading order)
```