Technical and numerical doc

Release 1.3

S.Jullien, M.Caillaud, R. Benshila, L. Bordois ,

G. Cambon, F. Dufois, F. Dumas, J. Gula ,

M. Le Corre, S. Le Gac, S. Le Gentil ,

F. Lemarié, P. Marchesiello, G. Morvan ,

J. Pianezze, and S. Theetten

Nov 29, 2022

CONTENTS

1 Primitive Equations **3**
- 1.1 Equations in Cartesian coordinates . . . 3
- 1.2 Equations in terrain following coordinates . . . 4

2 Quasi-Hydrostatic Equations **7**
- 2.1 Equations in Cartesian coordinate . . . 7

3 Wave-averaged Equations **9**
- 3.1 Equations in Cartesian coordinates . . . 9
- 3.2 Embedded wave model . . . 10
- 3.3 Breaking acceleration and bottom streaming . . . 11
- 3.4 Formulation of wave energy dissipation . . . 12

4 Non-Hydrostatic, Non-Boussinesq Equations **15**

5 Model variables **17**
- 5.1 Domain variables (*grid.h*) . . . 17
- 5.2 Barotropic variables (*ocean2d.h*) . . . 18
- 5.3 Tri-dimensionnal variables (*ocean3d.h*) . . . 18
- 5.4 Surface forcing (forces.h) . . . 19

6 Grid and Coordinates **21**
- 6.1 Vertical Grid parameters . . . 21
- 6.2 Wetting-Drying . . . 24

7 Numerics **25**
- 7.1 Overview . . . 25
- 7.2 Time Stepping . . . 26
- 7.3 Advection Schemes . . . 30
- 7.4 Pressure gradient . . . 36
- 7.5 Equation of State . . . 36
- 7.6 Wetting and Drying . . . 37
- 7.7 Non-Boussinesq Solver . . . 37

8 Parametrizations **39**
- 8.1 Vertical mixing parametrizations . . . 39
- 8.2 Horizontal diffusion . . . 45
- 8.3 Bottom friction . . . 47

9 Parallelisation **49**
- 9.1 Parallel strategy overview . . . 49
- 9.2 Loops and indexes . . . 52
- 9.3 Exchanges . . . 54
- 9.4 Dealing with outputs . . . 54

10 Atmospheric Surface Boundary Layer **57**

11 Open boundaries conditions **61**
11.1 OBC . . . 61
11.2 Sponge Layer . . . 62
11.3 Nudging layers . . . 62
11.4 Lateral forcing . . . 62

12 Rivers **65**

13 Tides **67**

14 Nesting Capabilities **69**

15 Sediment and Biology models **71**
15.1 Bottom Boundary Layer model . . . 71
15.2 Sediment models . . . 74
15.3 Biogeochemical models . . . 120
15.4 Lagrangian floats . . . 121

16 Coupling CROCO with other models **123**
16.1 OASIS philosophy . . . 123
16.2 Detailed OASIS implementation . . . 127
16.3 Coupled variables . . . 131
16.4 Grids . . . 138

17 I/O and Online Diagnostics **141**

18 Review of test cases **143**
18.1 Basin . . . 143
18.2 Canyon . . . 143
18.3 Equator . . . 145
18.4 Inner Shelf . . . 147
18.5 River Runoff . . . 149
18.6 Gravitational/Overflow . . . 149
18.7 Seamount . . . 151
18.8 Shelf front . . . 152
18.9 Equatorial Rossby Wave . . . 153
18.10 Thacker . . . 154
18.11 Upwelling . . . 154
18.12 Baroclinic Vortex . . . 155
18.13 Internal Tide . . . 157
18.14 Internal Tide (COMODO) . . . 157
18.15 Baroclinic Jet . . . 159
18.16 Plannar Beach . . . 160
18.17 Rip Current . . . 163
18.18 Sandbar . . . 165
18.19 Swash . . . 168
18.20 Tank . . . 170
18.21 Acoustic wave . . . 171
18.22 Gravitational Adjustment . . . 173
18.23 Internal Soliton . . . 174
18.24 Kelvin-Helmoltz Instability . . . 176
18.25 Horizontal tracer advection . . . 177
18.26 Sediment test cases . . . 178

19 Appendices **195**
19.1 cppdefs.h . . . 195
19.2 croco.in . . . 204
19.3 Comparison of ROMS and CROCO versions . . . 210

Related CPP options:

SOLVE3D	solve 3D primitive equations
UV_COR	Activate Coriolis terms
UV_ADV	Activate advection terms
NBQ	Activate non-boussinesq option
CROCO_QH	Activate quasi-hydrostatique option
MRL_WCI	Activate wave-current interactions

Preselected options:

```
# define SOLVE3D
# define UV_COR
# define UV_ADV
# undef  NBQ
# undef  CROCO_QH
# undef  MRL_WCI
```

Presentation

By default (#undef NBQ), CROCO solves the primitive equations as in ROMS, from which it inherited the robustness and efficiency of its time-splitting implementation (Shchepetkin & McWilliams, 2005; Debreu et al., 2012) and the NBQ option proposes an extension for nonhydrostatic applications. In CROCO's time-splitting algorithm, the "slow mode" is similar to ROMS internal (baroclinic) mode described in Shchepetkin & McWilliams (2005), whereas, the "fast mode" can include, in addition to the external (barotropic) mode, the pseudo-acoustic mode that allows computation of the nonhydrostatic pressure within a non-Boussinesq approach (Auclair et al., 2018). In this case, the slow internal mode is also augmented by a prognostic equation of vertical velocity, replacing the hydrostatic equation. Another option (CROCO_QH) extends the PE equations to form the quasi-hydrostatic equations, relaxing the hypothesis of weak horizontal Coriolis force (Marshall et al., 1997), thus adding a non-hydrostatic pressure component that is solved diagnostically. Then another option (MRL_WCI) treats the wave-averaged equations (McWilliams et al., 2004) with wave-current interaction terms that are both conservative and non-conservative (needing parametrizations).

CHAPTER

ONE

PRIMITIVE EQUATIONS

At resolutions larger than 1 km (more marginally above 100 m), The ocean is a fluid that can be described to a good approximation by the primitive equations. The PE equations are simplifications from the Navier-Stokes equations made from scale considerations, along with a nonlinear equation of state, which couples the two active tracers (temperature and salinity):

- Hydrostatic hypothesis: the vertical momentum equation is reduced to a balance between the vertical pressure gradient and the buoyancy force (nonhydrostatic processes such as convection must be parametrized)
- Boussinesq hypothesis: density variations are neglected except in their contribution to the buoyancy force
- Incompressibility hypothesis (steming from the former): the three dimensional divergence of the velocity vector is assumed to be zero.
- Spherical earth approximation: the geopotential surfaces are assumed to be spheres so that gravity (local vertical) is parallel to the earth's radius
- Thin-shell approximation: the ocean depth is neglected compared to the earth's radius
- Turbulent closure hypothesis: the turbulent fluxes (which represent the effect of small scale processes on the large-scale) are expressed in terms of large-scale features

By default (#undef NBQ), CROCO solves the primitive equations. But it has also the ability to relax the first 3 hypothesis (#define NBQ). When SOLV3D is not activated, CROCO can be used as a classical shallow water model.

1.1 Equations in Cartesian coordinates

- The momentum balance in zonal x and meridional y directions, written in terms of grid-scale (resolved) and subgrid-scale velocity components:

$$\frac{\partial u}{\partial t} + \vec{\nabla}.(\vec{\mathbf{v}}u) - fv = -\frac{\partial \phi}{\partial x} + \mathcal{F}_u + \mathcal{D}_u$$
$$\frac{\partial v}{\partial t} + \vec{\nabla}.(\vec{\mathbf{v}}v) + fu = -\frac{\partial \phi}{\partial y} + \mathcal{F}_v + \mathcal{D}_v$$

Turbulent closure schemes are applied to parametrized subgrid-scale vertical fluxes.

- The time evolution of a scalar concentration field, $C(x, y, z, t)$ (e.g. salinity, temperature, or nutrients), is governed by the advective-diffusive equation :

$$\frac{\partial C}{\partial t} + \vec{\nabla}.(\vec{\mathbf{v}}C) = \mathcal{F}_C + \mathcal{D}_C$$

- The equation of state is given by :

$$\rho = \rho(T, S, P)$$

- In the Boussinesq approximation, density variations are neglected in the momentum equations except in their contribution to the buoyancy force in the vertical momentum equation. Under the hydrostatic approximation, it is further assumed that the vertical pressure gradient balances the buoyancy force :

$$\frac{\partial \phi}{\partial z} = -\frac{\rho g}{\rho_0}$$

- The final equation expresses the continuity equation. For an incompressible fluid (Boussinesq approximation):

$$\vec{\nabla}.\vec{\mathbf{v}} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

The variables used are :

$\mathcal{D}_u, \mathcal{D}_v, \mathcal{D}_C$: diffusive terms

$\mathcal{F}_u, \mathcal{F}_v, \mathcal{F}_C$: forcing terms

$f(x, y)$: Traditional Coriolis parameter $2\Omega sin\phi$

g : acceleration of gravity

$\phi(x, y, z, t)$: dynamic pressure $\phi = P/\rho_0$, with P the total pressure

$\rho_0 + \rho(x, y, z, t)$: total in situ density

u, v, w : the (x,y,z) components of vector velocity $\vec{\mathbf{v}}$

1.2 Equations in terrain following coordinates

We first introduce a generalized stretched vertical coordinate system (s), which sets the variable bottom flat at $z = -h(x, y)$. s spans the range from -1 (bottom) to 0 (surface) and the transformation rules are:

$$\left(\frac{\partial}{\partial x}\right)_z = \left(\frac{\partial}{\partial x}\right)_s - \left(\frac{1}{H_z}\right)\left(\frac{\partial z}{\partial x}\right)_s \frac{\partial}{\partial s}$$
$$\left(\frac{\partial}{\partial y}\right)_z = \left(\frac{\partial}{\partial y}\right)_s - \left(\frac{1}{H_z}\right)\left(\frac{\partial z}{\partial y}\right)_s \frac{\partial}{\partial s}$$
$$\frac{\partial}{\partial z} = \left(\frac{\partial s}{\partial z}\right)\frac{\partial}{\partial s} = \frac{1}{H_z}\frac{\partial}{\partial s}$$
$$\text{where } H_z \equiv \frac{\partial z}{\partial s}$$

The vertical velocity in s coordinate is:

$$\Omega(x, y, s, t) = \frac{1}{H_z}\left[w - (1+s)\frac{\partial \zeta}{\partial t} - u\frac{\partial z}{\partial x} - v\frac{\partial z}{\partial y}\right]$$
$$w = \frac{\partial z}{\partial t} + u\frac{\partial z}{\partial x} + v\frac{\partial z}{\partial y} + \Omega H_z$$

$\Omega = 0$ at both surface and bottom.

Next, the requirement for a laterally variable grid resolution can also be met, for suitably smooth domains, by introducing an appropriate orthogonal coordinate transformation in the horizontal. Let the new coordinates be $\xi(x, y)$ and $\eta(x, y)$ where the relationship of horizontal arc length to the differential distance is given by:

$$(ds)_\xi = \left(\frac{1}{m}\right) d\xi$$
$$(ds)_\eta = \left(\frac{1}{n}\right) d\eta$$

Here $m(\xi,\eta)$ and $n(\xi,\eta)$ are the scale factors which relate the differential distances $(\Delta\xi,\Delta\eta)$ to the actual (physical) arc lengths.

$$
\begin{aligned}
\frac{\partial}{\partial t}\left(\frac{H_z u}{mn}\right)+\frac{\partial}{\partial \xi}\left(\frac{H_z u^2}{n}\right)+\frac{\partial}{\partial \eta}\left(\frac{H_z uv}{m}\right)+\frac{\partial}{\partial s}\left(\frac{H_z u\Omega}{mn}\right)& \\
-\left\{\left(\frac{f}{mn}\right)+v\frac{\partial}{\partial \xi}\left(\frac{1}{n}\right)-u\frac{\partial}{\partial \eta}\left(\frac{1}{m}\right)\right\}H_z v=& \\
-\left(\frac{H_z}{n}\right)\left(\frac{\partial \phi}{\partial \xi}+\frac{g\rho}{\rho_o}\frac{\partial z}{\partial \xi}+g\frac{\partial \zeta}{\partial \xi}\right)+\frac{H_z}{mn}\left(\mathcal{F}_u+\mathcal{D}_u\right)& \\
\frac{\partial}{\partial t}\left(\frac{H_z v}{mn}\right)+\frac{\partial}{\partial \xi}\left(\frac{H_z uv}{n}\right)+\frac{\partial}{\partial \eta}\left(\frac{H_z v^2}{m}\right)+\frac{\partial}{\partial s}\left(\frac{H_z v\Omega}{mn}\right)& \\
+\left\{\left(\frac{f}{mn}\right)+v\frac{\partial}{\partial \xi}\left(\frac{1}{n}\right)-u\frac{\partial}{\partial \eta}\left(\frac{1}{m}\right)\right\}H_z u=& \\
-\left(\frac{H_z}{m}\right)\left(\frac{\partial \phi}{\partial \eta}+\frac{g\rho}{\rho_o}\frac{\partial z}{\partial \eta}+g\frac{\partial \zeta}{\partial \eta}\right)+\frac{H_z}{mn}\left(\mathcal{F}_v+\mathcal{D}_v\right)& \\
\frac{\partial}{\partial t}\left(\frac{H_z T}{mn}\right)+\frac{\partial}{\partial \xi}\left(\frac{H_z uT}{n}\right)+\frac{\partial}{\partial \eta}\left(\frac{H_z vT}{m}\right)+\frac{\partial}{\partial s}\left(\frac{H_z \Omega T}{mn}\right)=\frac{H_z}{mn}\left(\mathcal{F}_T+\mathcal{D}_T\right)& \\
\frac{\partial}{\partial t}\left(\frac{H_z S}{mn}\right)+\frac{\partial}{\partial \xi}\left(\frac{H_z uS}{n}\right)+\frac{\partial}{\partial \eta}\left(\frac{H_z vS}{m}\right)+\frac{\partial}{\partial s}\left(\frac{H_z \Omega S}{mn}\right)=\frac{H_z}{mn}\left(\mathcal{F}_S+\mathcal{D}_S\right)& \\
\rho=\rho(T,S,P)& \\
\frac{\partial \phi}{\partial s}=-\left(\frac{gH_z\rho}{\rho_o}\right)& \\
\frac{\partial}{\partial t}\left(\frac{H_z}{mn}\right)+\frac{\partial}{\partial \xi}\left(\frac{H_z u}{n}\right)+\frac{\partial}{\partial \eta}\left(\frac{H_z v}{m}\right)+\frac{\partial}{\partial s}\left(\frac{H_z \Omega}{mn}\right)=0&
\end{aligned}
$$

CHAPTER

TWO

QUASI-HYDROSTATIC EQUATIONS

In oceanography, traditional approximation (TA) takes the Coriolis force only partially into account by neglecting the components proportional to the cosine of latitude: $\tilde{f} = 2\Omega cos\phi$ (see Gerkema et al., 2008, for a review). The justification for the TA is in the hypothesis that the depth of the oceans is very thin compared to the radius of the Earth. The vertical motions must then be much weaker than the horizontal ones, rendering the non-tradiional (NT) Coriolis terms (with $\tilde{f}$) insignificant compared to the traditional terms (with f) and rendering the pressure field nearly hydrostatic. Similarly, strong vertical stratification in density, which suppresses vertical motions, also diminishes the role of NT terms. However, this argument becomes weak near the equator ($\tilde{f} >> f$), or in motions with a strong vertical component (e.g., convection).

Note also that the QH momentum equations are shown to be more dynamically consistent than PE hydrostatic equations and that they correctly imply conservation laws for energy, angular momentum, and potential vorticity

2.1 Equations in Cartesian coordinate

- The momentum balance in x and y directions is extended to include $\tilde{f}$ terms (zonal u component):

$$\frac{\partial u}{\partial t} + \vec{\nabla}.(\vec{\mathbf{v}}u) - fv + \tilde{f}w = -\frac{\partial \phi}{\partial x} + \mathcal{F}_u + \mathcal{D}_u$$
$$\frac{\partial v}{\partial t} + \vec{\nabla}.(\vec{\mathbf{v}}v) + fu = -\frac{\partial \phi}{\partial y} + \mathcal{F}_v + \mathcal{D}_v$$

- Under the QH approximation, the quasi-hydrostatic balance is used for the vertical momentum equation, where the zonal flow partially balances the pressure gradient :

$$\frac{\partial \phi}{\partial z} = -\frac{\rho g}{\rho_0} + \tilde{f}u$$

In practice, the non-traditional term $\tilde{f}u$ is introduce as a correction to density (in the density computation subroutine rho_eos).

The variables used are :

$\mathcal{D}_u, \mathcal{D}_v$: diffusive terms

$\mathcal{F}_u, \mathcal{F}_v$: forcing terms

$f(x, y)$: Traditional Coriolis parameter $2\Omega sin\phi$

$\tilde{f}(x, y)$: Non-traditional Coriolis parameter $2\Omega cos\phi$

g : acceleration of gravity

$\phi(x, y, z, t)$: dynamic pressure $\phi = P/\rho_0$, with P the total pressure

$\rho_0 + \rho(x, y, z, t)$: total in situ density

u, v, w : the (x,y,z) components of vector velocity $\vec{\mathbf{v}}$

CHAPTER

THREE

WAVE-AVERAGED EQUATIONS

MRL_WCI	Activate wave-current interactions
MRL_CEW	Activate current effect on waves (2-way interaction)
ANA_WWAVE	Analytical (constant) wave parameters (Hs,Tp,Dir)
WAVE_OFFLINE	Activate wave forcing from offline model/data
WKB_WWAVE	Activate CROCO's monochromatic (WKB) model
OW_COUPLING	Activate coupling with spectral wave model (WW3)
WAVE_FRICTION	Activate bottom friction for WKB model and WAVE_STREAMING
WAVE_STREAMING	Activate bottom streaming (needs WAVE_FRICTION)
STOKES_DRIFT	Activate Stokes drift

Preselected options:

```
# define STOKES_DRIFT
```

A vortex-force formalism for the interaction of surface gravity waves and currents is implemented in CROCO (Marchesiello et al., 2015; Uchiyama et al., 2010). Eulerian wave-averaged current equations for mass, momentum, and tracers are included based on an asymptotic theory by McWilliams et al. (2004) plus non-conservative wave effects due to wave breaking, associated surface roller waves, bottom streaming, and wave-enhanced vertical mixing and bottom drag especially for coastal and nearshore applications. The wave information is provided by either a spectrum-peak WKB wave-refraction model that includes the effect of currents on waves, or, alternatively, a spectrum-resolving wave model (e.g., WAVEWATCH3) can be used. In nearshore applications, the currents' cross-shore and vertical structure is shaped by the wave effects of near-surface breaker acceleration, vertical component of vortex force, and wave-enhanced pressure force and bottom drag.

3.1 Equations in Cartesian coordinates

In the Eulerian wave-averaged current equations, terms for the wave effect on currents (WEC) are added to the primitive equations. Three new variables are defined:

$$\xi^c = \xi + \hat{\xi}$$
$$\phi^c = \phi + \hat{\phi}$$
$$\vec{\mathbf{v}}_L = \vec{\mathbf{v}} + \vec{\mathbf{v}}_S$$

where ξ^c is a composite sea level, ϕ^c absorbs the Bernoulli head $\hat{\phi}$, $\vec{\mathbf{v}_L}$ is the wave-averaged Lagrangian velocity, sum of Eulerian velocity and Stokes drift $\vec{\mathbf{v}_S}$. The 3D Stokes velocity is non-divergent and defined for a

monochromatic wave field (amplitude A, wavenumber vector $\vec{\mathbf{k}} = (k_x, k_y)$, and frequency σ) by:

$$u_S = \frac{A^2\sigma}{2\sinh^2(kD)}\cosh(2k(z+h))\,k_x$$
$$v_S = \frac{A^2\sigma}{2\sinh^2(kD)}\cosh(2k(z+h))\,k_y$$
$$w_S = -\int_{-h}^{z}\left(\frac{\partial u_S}{\partial x}+\frac{\partial v_S}{\partial y}\right)dz'$$

Where $D = h + \xi^c$. The quasi-static sea level and Bernouilli head are:

$$\hat{\xi} = -\frac{A^2 k}{2\sinh(2kD)}$$
$$\hat{\phi} = \frac{A^2\sigma}{4k\sinh^2(kD)}\int_{-h}^{z}\frac{\partial^2\vec{\mathbf{k}}.\vec{\mathbf{v}}}{\partial z'^2}\sinh(2k(z-z'))\;dz'$$

The primitive equations become (after re-organizing advection and vortex force terms):

$$\frac{\partial u}{\partial t}+\vec{\nabla}.(\vec{\mathbf{v}}_L u)-fv_L = -\frac{\partial\phi^c}{\partial x}+\left(u_S\frac{\partial u}{\partial x}+v_S\frac{\partial v}{\partial x}\right)+\mathcal{F}_u+\mathcal{D}_u+\mathcal{F}^{\mathcal{W}}{}_u$$
$$\frac{\partial v}{\partial t}+\vec{\nabla}.(\vec{\mathbf{v}}_L v)+fu_L = -\frac{\partial\phi^c}{\partial y}+\left(u_S\frac{\partial u}{\partial y}+v_S\frac{\partial v}{\partial y}\right)+\mathcal{F}_v+\mathcal{D}_v+\mathcal{F}^{\mathcal{W}}{}_v$$
$$\frac{\partial\phi^c}{\partial z}+\frac{\rho g}{\rho_0} = \vec{\mathbf{v}}_S.\frac{\partial\vec{\mathbf{v}}}{\partial z}$$
$$\frac{\partial C}{\partial t}+\vec{\nabla}.(\vec{\mathbf{v}}_L C) = \mathcal{F}_C+\mathcal{D}_C+\mathcal{F}^{\mathcal{W}}{}_C$$
$$\vec{\nabla}.\vec{\mathbf{v}}_L = 0$$
$$\rho = \rho(T,S,P)$$

The variables used are :

$\mathcal{D}_u, \mathcal{D}_v, \mathcal{D}_C$: diffusive terms (including wave-enhaced bottom drag and mixing)

$\mathcal{F}_u, \mathcal{F}_v, \mathcal{F}_C$: forcing terms

$\mathcal{F}^{\mathcal{W}}{}_u, \mathcal{F}^{\mathcal{W}}{}_v, \mathcal{F}^{\mathcal{W}}{}_C$: wave forcing terms (bottom streaming, breaking acceleration)

$f(x,y)$: Traditional Coriolis parameter $2\Omega sin\phi$

g : acceleration of gravity

$\phi(x,y,z,t)$: dynamic pressure $\phi = P/\rho_0$, with P the total pressure

$\rho_0 + \rho(x,y,z,t)$: total in situ density

u, v, w : the (x,y,z) components of vector velocity $\vec{\mathbf{v}}$

3.2 Embedded wave model

WKB_WWAVE	Activate WKB wave model
WAVE_ROLLER	Activate wave rollers
WAVE_FRICTION	Activate bottom friction
WKB_ADD_DIFF	Activate additional diffusion to wavenumber field
MRL_CEW	Active current effect on waves
WKB_KZ_FILTER	Activate space filter on ubar,vbar,zeta for CEW
WKB_TIME_FILTER	Activate time filter on ubar,vbar,zeta for CEW
WAVE_RAMP	Activate wave ramp
ANA_BRY_WKB	Read boundary data from croco.in
WKB_OBC_WEST	Offshore wave forcing at western boundary
WKB_OBC_EAST	Offshore wave forcing at eastern boundary

Preselected options:

```
# ifdef MRL_CEW
#  undef  WKB_KZ_FILTER
#  undef  WKB_TIME_FILTER
# endif
# define WKB_ADD_DIFF
# if defined SHOREFACE || defined SANDBAR || (defined RIP && !defined BISCA)
#  define ANA_BRY_WKB
# endif
```

A WKB wave model for monochromatic waves is embedded in CROCO following Uchiyama et al. (2010). It is based on the conservation of wave action $\mathcal{A} = E/\sigma$ and wavenumber $\mathbf{k}$ – wave crest conservation – and is particularly suitable for nearshore beach applications, allowing refraction from bathymetry and currents (but no diffraction or reflection), with parametrizations for wave breaking and bottom drag:

$$\frac{\partial \mathcal{A}}{\partial t} + \vec{\nabla}.\mathcal{A}\vec{\mathbf{c}}_g = -\frac{\epsilon^w}{\sigma}$$

$$\frac{\partial \vec{\mathbf{k}}}{\partial t} + \vec{\mathbf{c}}_g.\nabla\vec{\mathbf{k}} = -\vec{\mathbf{k}}.\nabla\vec{\mathbf{V}} - \frac{k\sigma}{\sinh 2kD}\nabla D$$

$\vec{\mathbf{V}}$ is the depth-averaged velocity vector and σ is the intrinsic frequency defined by the linear dispersion relation $\sigma^2 = gk\tanh kD$. Current effects on waves are noticeable in the groupe velocity c_g which gets two components: the doppler shift due to currents on waves and the groupe velocity of the primary carrier waves :

$$\vec{\mathbf{c}}_g = \vec{\mathbf{V}} + \frac{\sigma}{2k^2}\left(1 + \frac{2kD}{\sinh 2kD}\right)\vec{\mathbf{k}}$$

The currents may need filtering before entering the wave model equations because the current field should evolve slowly with respect to waves in the asymptotic regime described by McWilliams et al. (2004). By default, this filtering is turned off (WKB_KZ_FILTER, WKB_TIME_FILTER).

ϵ^w is the depth-integrated rate of wave energy dissipation due to depth-induced breaking ϵ^b (including white capping) and bottom friction ϵ^{wd}, both of which must be parameterized (in WKB, WW3 or CROCO if defined WAVE_OFFLINE):

$$\epsilon^w = \epsilon^b + \epsilon^{wd}$$

3.3 Breaking acceleration and bottom streaming

A formulation for ϵ^b is needed in both the wave model (dissipation term) and the circulation model (acceleration term). In the wave-averaged momentum equations of the circulation model, the breaking acceleration enters as a body force through $\mathcal{F}^{\mathcal{W}}$:

$$\vec{\mathbf{F}^{\mathbf{b}}} = \frac{\epsilon^b}{\rho\sigma}\vec{\mathbf{k}}\, f_b(z)$$

where $f_b(z)$ is a normalized vertical distribution function representing vertical penetration of momentum associated with breaking waves from the surface. The penetration depth is controlled by a vertical length-scale taken as H_{rms}.

The wave model can also include a roller model with dissipation ϵ^r. In this case:

$$\vec{\mathbf{F}^{\mathbf{b}}} = \frac{(1-\alpha_r)\epsilon^b + \epsilon^r}{\rho\sigma}\vec{\mathbf{k}}\, f_b(z)$$

The idea is that some fraction α_r of wave energy is converted into rollers that propagate toward the shoreline before dissipating, while the remaining fraction $1-\alpha_r$ causes local dissipation (hence current acceleration). It can be useful for correcting ϵ^b with some flexibility to depict different breaking wave and beach forms (e.g., spilling or

plunging breakers, barred or plane beaches), although the parameter B_b can also be used for that. See Uchiyama et al. (2010) for the roller equation and ϵ^r formulation.

Wave-enhanced bottom dissipation enters in the momentum equations through a combined wave-current drag formulation (see parametrizations) and bottom streaming. The latter is due to dissipation of wave energy in the wave boundary layer that causes the instantaneous, oscillatory wave bottom orbital velocities to be slightly in phase from quadrature; this causes a wave stress (bottom streaming) in the wave bottom boundary layer along the direction of wave propagation (Longuet-Higgins, 1953). The effect of bottom streaming in momentum balance is accounted for by using the wave dissipation due to bottom friction with an upward decaying vertical distribution:

$$\vec{\mathbf{F}^{\mathbf{st}}} = \frac{\epsilon^{wd}}{\rho\sigma}\vec{\mathbf{k}}\, f_{st}(z)$$

where $f_{st}(z)$ is a vertical distribution function.

3.4 Formulation of wave energy dissipation

WAVE_SFC_BREAK	Activate surface breaking acceleration
WAVE_BREAK_CT93	Activate Church & Thorton (1993) breaking acceleration (default)
WAVE_BREAK_TG86	Activate Thorton & Guza (1983, 1986)

Preselected options:

```
# define WAVE_BREAK_CT93
# undef  WAVE_BREAK_TG86
# undef  WAVE_SFC_BREAK
```

While a few formulations for ϵ^b are implemented in CROCO, the one by Church and Thornton (1993) is generally successful for nearshore beach applications:

$$\epsilon^b = \frac{3}{16}\sqrt{\pi}\rho g B_b^3 \frac{H_{rms}^3}{D}\left\{1+\tanh\left[8\left(\frac{H_{rms}}{\gamma_b D}-1\right)\right]\right\}\left\{1-\left[1+\left(\frac{H_{rms}}{\gamma_b D}\right)^2\right]^{-2.5}\right\}$$

where B_b and γ_b are empirical parameters related to wave breaking. γ_b represents the wave height-to-depth ratio for which all waves are assumed to be breaking and B_b is the fraction of foam on the face, accounting for the type of breaker. H_{rms} is the RMS wave height. For the DUCK94 experiment, Uchiyama et al. (2010) suggest $\gamma_b = 0.4$ and $B_b = 0.8$, while for Biscarrosse Beach, Marchesiello et al. (2015) use $\gamma_b = 0.3$ and $B_b = 1.3$ from calibration with video cameras.

For ϵ^{wd}, the dissipation caused by bottom viscous drag on the primary waves, we use a parameterization for the realistic regime of a turbulent wave boundary layer, consistent with the WKB spectrum-peak wave modeling:

$$\epsilon^{wd} = \frac{1}{2\sqrt{\pi}}\rho f_w u_{orb}^3$$

where u_{orb} is the wave orbital velocity magnitude and f_w is a wave friction factor, function of roughness length z_0:

$$u_{orb} = \frac{\sigma H_{rms}}{2\sinh kD}$$
$$f_w = 1.39\left(\frac{\sigma z_0}{u_{orb}}\right)^{0.52}$$

References

Marchesiello, P.; Benshila, R.; Almar, R.; Uchiyama, Y.; McWilliams, J.C., and Shchepetkin, A., 2015. On tridimensional rip current modeling. Ocean Model., 96, 36-48.

McWilliams, J.C., Restrepo, J.M., and Lane, M.R., 2004. An asymptotic theory for the interaction of waves and currents in coastal waters. J. Fluid Mech., 511, 135–178.

Thornton, E.B. & R.T. Guza, 1983: Transformation of wave height distribution, J. Geophys. Res. 88, 5925-5938.

Uchiyama, Y., McWilliams, J., Shchepetkin, A., 2010. Wave-current interaction in an oceanic circulation model with a vortex-force formalism: application to the surf zone. Ocean Modell. 34, 16–35.

Weir, B., Uchiyama, Y., Lane, E. M., Restrepo, J. M., & McWilliams, J. C. (2011). A vortex force analysis of the interaction of rip currents and surface gravity waves. Journal of Geophysical Research: Oceans, 116(C5).

CHAPTER

FOUR

NON-HYDROSTATIC, NON-BOUSSINESQ EQUATIONS

The full set of Navier-Stokes equations for a free-surface ocean is explicitly integrated in the non-hydrostatic, non-Boussinesq version of CROCO (#define NBQ). In this approach, acoustic waves are solved explicitly to avoid Boussinesq-degeneracy, which inevitably leads to a 3D Poisson-system in non-hydrostatic Boussinesq methods – detrimental to computational costs and difficult to implement within a split-explicit barotropic/baroclinic model.

NBQ equations include the momentum and continuity equations, the surface kinematic relation (for free surface), temperature, salinity – or other tracer C – and the equation of state, which reads in Cartesian coordinates:

$$
\begin{aligned}
\frac{\partial \rho u}{\partial t} + \vec{\nabla}.(\rho \vec{\mathbf{v}} u) - \rho f v - \rho \tilde{f} w &= -\frac{\partial P}{\partial x} + \lambda \frac{\partial \vec{\nabla}.\vec{\mathbf{v}}}{\partial x} + \mathcal{F}_u + \mathcal{D}_u \\
\frac{\partial \rho v}{\partial t} + \vec{\nabla}.(\rho \vec{\mathbf{v}} v) + \rho f u &= -\frac{\partial P}{\partial y} + \lambda \frac{\partial \vec{\nabla}.\vec{\mathbf{v}}}{\partial y} + \mathcal{F}_v + \mathcal{D}_v \\
\frac{\partial \rho w}{\partial t} + \vec{\nabla}.(\rho \vec{\mathbf{v}} w) - \rho \tilde{f} u &= -\frac{\partial P}{\partial z} - \rho g + \lambda \frac{\partial (\vec{\nabla}.\vec{\mathbf{v}})}{\partial z} + \mathcal{F}_w + \mathcal{D}_w \\
\frac{\partial \rho}{\partial t} &= -\vec{\nabla}.(\rho \vec{\mathbf{v}}) \\
\frac{\partial \xi}{\partial t} &= w_f|_{z=\xi} - \vec{\mathbf{v}}|_{z=\xi}.\vec{\nabla}\xi \\
\frac{\partial \rho C}{\partial t} &= -\vec{\nabla}.(\rho \vec{\mathbf{v}} C) + \mathcal{F}_C + \mathcal{D}_C
\end{aligned}
$$

λ is the second (bulk) viscosity, associated with compressibility (it can be used to damp acoustic waves).

A relation between ρ and P is now required. To that end, and as part of a time-splitting approach, density is decomposed into slow and fast components based on a first-order decomposition with respect to total pressure. In the following, s and f subscripts refer to slow and fast-mode components respectively:

$$
\rho = \rho_s(T,S,P) + \overbrace{\left.\frac{\partial \rho}{\partial P}\right|_{T,S}}^{\rho_f = c_s^{-2} P_f} \delta P + O(\delta P^2)
$$

$$
P = \underbrace{P_{atm} + \int_z^\xi (\rho_s - \rho_0) g \, dz'}_{SLOW} + \underbrace{\rho_0 g(\xi - z) + \overbrace{\delta P}^{P_f}}_{FAST}
$$

c_s is the speed of sound and $\delta P = P_f$ is the nonhydrostatic pressure.

The Navier-Stokes equations are then integrated with two different time-steps within the time-splitting approach. The slow mode is identical to ROMS whereas the fast mode (in the NBQ equations) is 3D and the fast time step

includes the integration of the compressible terms of the momentum and continuity equations. In vector form:

$$\frac{\partial \rho \vec{\mathbf{v}}}{\partial t} = \underbrace{-\vec{\nabla}.(\rho \vec{\mathbf{v}} \otimes \vec{\mathbf{v}}) - 2\rho\vec{\Omega} \times \vec{\mathbf{v}} - \vec{\nabla}(\int_z^{\xi_f} (\rho_s - \rho_0) g \, dz') + \vec{\mathcal{F}}_{\vec{\mathbf{v}}} + \vec{\mathcal{D}}_{\vec{\mathbf{v}}}}_{SLOW}$$

$$\underbrace{-\rho_0 g \vec{\nabla}\xi_f - \vec{\nabla} P_f + \rho \vec{\mathbf{g}} + \lambda \vec{\nabla}(\vec{\nabla}.\vec{\mathbf{v}})}_{FAST}$$

$$\frac{\partial \rho_f}{\partial t} = -\frac{\partial \rho_s}{\partial t} - \vec{\nabla}.(\rho \vec{\mathbf{v}})$$

$$P_f = c_s^2 \, \rho_f$$

$$\frac{\partial \xi_f}{\partial t} = w_f|_{z=\xi} - \vec{\mathbf{v}}_f|_{z=\xi}.\vec{\nabla}\xi_f$$

$$\frac{\partial \rho C_s}{\partial t} = -\vec{\nabla}.(\rho \vec{\mathbf{v}} C_s) + \mathcal{F}_C + \mathcal{D}_C$$

$$\rho_s = \rho(T_s, S_s, \xi_f)$$

$$\rho = \rho_s + \rho_f$$

The momentum is integrated both in slow and fast modes but the right-hand-side of the equation is split in two parts: a slow part, made of slowly varying terms (advection, Coriolis force, baroclinic pressure force and viscous dissipation), and a fast part, made of fast-varying terms (the surface-induced and compressible pressure force, the weight and the dissipation associated with bulk-viscosity). This momentum equation is numerically integrated twice, once with a large time-step keeping the fast part constant, and once with a smaller time-step keeping the slow part constant. This is much more computationally efficient than integrating the whole set of equations at the same fast time step. More details can be found in Auclair et al. (2018).

Note that the solved acoustic waves can become pseudo-acoustic if their phase speed c_s is artificially slowed down (it is a model input). In this case, high-frequency processes associated with bulk compressibility may be unphysical, but a coherent solution for slow non-hydrostatic dynamics is preserved, while the CFL constraint is relaxed.

CHAPTER

FIVE

MODEL VARIABLES

Model variables are defined in .h Fortran 77 files :

5.1 Domain variables (*grid.h*)

grid.h : Environmental two-dimensional arrays associated with curvilinear horizontal coordinate system

h : Model topography (bottom depth [m] at RHO-points.)
dh : Topograhy increment in case of moving bathymetry
f : Coriolis parameter [1/s].
fomn : Compound term, f/[pm*pn] at RHO points.

angler : Angle [radians] between XI-axis and the direction to the EAST at RHO-points.

latr : Latitude (degrees_north) at RHO-, U-, and V-points.
latu
latv
lonr : Longitude (degrees_east) at RHO-, U-, and V-points.
lonu
lonv

xp : XI-coordinates [m] at PSI-points.
xr : XI-coordinates (m] at RHO-points.
yp : ETA-coordinates [m] at PSI-points.
yr : ETA-coordinates [m] at RHO-points.

pm : Coordinate transformation metric “m” [1/meters] associated with the differential distances in XI.
pn : Coordinate transformation metric “n” [1/meters]associated with the differential distances in ETA.
om_u : Grid spacing [meters] in the XI -direction at U-points.
om_v : Grid spacing [meters] in the XI -direction at V-points.
on_u : Grid spacing [meters] in the ETA-direction at U-points.
on_v : Grid spacing [meters] in the ETA-direction at V-points.

dmde : ETA-derivative of inverse metric factor “m”, d(1/M)/d(ETA).
dndx : XI-derivative of inverse metric factor “n”, d(1/N)/d(XI).

pmon_p : Compound term, pm/pn at PSI-points.
pmon_r : Compound term, pm/pn at RHO-points.
pmon_u : Compound term, pm/pn at U-points.
pnom_p : Compound term, pn/pm at PSI-points.
pnom_r : Compound term, pn/pm at RHO-points.
pnom_v : Compound term, pn/pm at V-points.

rmask : Land-sea masking arrays at RHO-,U-,V- and PSI-points (rmask,umask,vmask) = (0=Land, 1=Sea)
umask
vmask
pmask : pmask=(0=Land, 1=Sea, 1-gamma2 =boundary).

reducu : reduction coefficient along x-axis for rivers sections
reducv : reduction coefficient along y-axis for rivers sections

5.2 Barotropic variables (*ocean2d.h*)

ocean2d.h : 2D dynamical variables for fast mode

zeta,rzeta : Free surface elevation [m] and its time tendency;
ubar,rubar : Vertically integrated 2D velocity components in
vbar,rvbar : XI- and ETA-directions and their time tendencies;

5.3 Tri-dimensionnal variables (*ocean3d.h*)

ocean2d.h : 3D tracers dynamical variables for baroclinic mode

u,v : 3D velocity components in XI- and ETA-directions
t : tracer array (temperature, salinity, passive tracers, sediment)
Hz : level thickness
z_r : depth at rho point
z_w : depth at w point
Huon : transport a U point
Hvon : transport at V point
We, Wi : vertical velocity (explicit, implicit)
rho : density anomaly
rho1 : potential density at 1 atm

5.4 Surface forcing (forces.h)

forces.h :

Surface momemtum flux (wind stress) :

sustr : XI- and ETA-components of kinematic surface momentum flux
svstr : wind stresses) defined at horizontal U- and V-points.dimensioned as [m^2/s^2].

Bottom mometum flux :

bustr : XI- and ETA-components of kinematic bottom momentum flux
bvstr : (drag) defined at horizontal U- and V-points [m^2/s^2].!

Surface tracers fluxes :

stflx : Kinematic surface fluxes of tracer type variables at horizontal RHO-points. Physical dimensions [degC m/s] - temperature; [PSU m/s] - salinity.
dqdt : Kinematic surface net heat flux sensitivity to SST [m/s].
sst : Current sea surface temperature [degree Celsius].
dqdtg : Two-time-level grided data for net surface heat flux
sstg : sensitivity to SST grided data [Watts/m^2/Celsius] and sea surface temperature [degree Celsius].
dqdtp : Two-time-level point data for net surface heat flux
sstp : sensitivity to SST grided data [Watts/m^2/Celsius] and sea surface temperature [degree Celsius].
tsst : Time of sea surface temperature data.
sss : Current sea surface salinity [PSU].
tair : surface air temperature at 2m [degree Celsius].
wsp : wind speed at 10m [degree Celsius].
rhum : surface air relative humidity 2m [fraction]
prate : surface precipitation rate [cm day-1]
radlw : net terrestrial longwave radiation [Watts meter-2]
radsw : net solar shortwave radiation [Watts meter-2]
patm2d : atmospheric pressure above mean seal level
paref : reference pressure to compute inverse barometer effect
srflx : Kinematic surface shortwave solar radiation flux [degC m/s] at horizontal RHO-points

Wind induced waves everything is defined at rho-point :

wfrq : wind-induced wave frequency [rad/s]
uorb : xi-component of wave-induced bed orbital velocity [m/s]
vorb : eta-component of wave-induced bed orbital velocity [m/s]
wdrx : cosine of wave direction [non dimension]
wdre : sine of wave direction [non dimension]
whrm : (RMS) wave height (twice the wave amplitude) [m]
wepb : breaking dissipation rate (epsilon_b term) [m3/s3]
wepd : frictional dissipation rate (epsilon_d term) [m3/s3]
wepr :roller dissipation rate (epsilon_r term) [m3/s3]

wbst : frictional dissipation stress (e_d k/sigma) [m2/s2]

Wave averaged quantities :

brk2dx : xi-direciton 2D breaking dissipation (rho)
brk2de : eta-direction 2D breaking dissipation (rho)
frc2dx : xi-direciton 2D frictional dissipation (rho)
frc2de : eta-direction 2D frictional dissipation (rho)
ust2d : xi-direciton Stokes transport (u-point)
vst2d : eta-direciton Stokes transport (v-point)
sup : quasi-static wave set-up (rho-point)
calP : pressure correction term (rho-point)
Kapsrf : Bernoulli head terrm at the surface (rho-point)
brk3dx : xi-direciton 3D breaking dissipation (rho)
brk3de : eta-direction 3D breaking dissipation (rho)
ust : xi-direciton 3D Stokes drift velocity (u-point)
vst : eta-direciton 3D Stokes drift velocity (v-point)
wst : vertical 3D Stokes drift velocity (rho-point)
Kappa : 3D Bernoulli head term (rho-point)
kvf : vertical vortex force term (K term, 3D, rho-point)
Akb : breaking-wave-induced additional diffusivity (w-point)
Akw : wave-induced additional diffusivity (rho-point)
E_pre : previous time-step value for Akw estimation (rho)
frc3dx : xi-direciton 3D frictional dissipation (rho)
frc3de : eta-direction 3D frictional dissipation (rho)

CHAPTER

SIX

GRID AND COORDINATES

Related CPP options:

CURVGRID	Activate curvilinear coordinate transformation
SPHERICAL	Activate longitude/latitude grid positioning
MASKING	Activate land masking
WET_DRY	Activate wetting-Drying scheme
NEW_S_COORD	Choose new vertical S-coordinates

Preselected options:

```
# define CURVGRID
# define SPHERICAL
# define MASKING
# undef WET_DRY
# undef NEW_S_COORD
```

6.1 Vertical Grid parameters

Two vertical transformation are available for the generalized vertical terrain following vertical system : By default, we have :

$$z(x,y,\sigma,t) = z_0(x,y,\sigma) + \zeta(x,y,t)\left[1 + \frac{z_0(x,y,\sigma)}{h(x,y)}\right] \tag{6.1}$$

$$z_0(x,y,\sigma) = h_c\sigma + [h(x,y) - h_c]\, Cs(\sigma) \tag{6.2}$$

When activated the cpp key NEW_S_COORD, we have :

$$z(x,y,\sigma,t) = \zeta(x,y,\sigma) + [\zeta(x,y,t) + h(x,y)]\, z_0(x,y,\sigma) \tag{6.3}$$

$$z_0(x,y,\sigma) = \frac{h_c\sigma + h(x,y)Cs(\sigma)}{h_c + h(x,y)} \tag{6.4}$$

with :

- $z_0(x,y,\sigma)$ a nonlinear vertical transformation
- $\zeta(x,y,\sigma)$ the free-surface
- $h(x,y)$ the ocean bottom
- σ afractional vertical stretching coordinate, $-1 \leq \sigma \leq 0$
- h_c a positive thickness controlling the stretching
- $Cs(\sigma)$ a nondimensional, monotonic, vertical stretching, $-1 \leq (C\sigma) \leq 0$

Vertical grid stretching is controlled by the following parameters, that have to be set similarly in `croco.in`, and `crocotools_param.m`:

theta_s	Vertical S-coordinate surface stretching parameter. When building the climatology and initial CROCO files, we have to define the vertical grid. Warning! The different vertical grid parameters should be identical in this crocotools_param.m and in croco.in. This is a serious cause of bug.
theta_b	Vertical S-coordinate bottom stretching parameter.
hc	Vertical S-coordinate Hc parameter. It gives approximately the transition depth between the horizontal surfacelevels and the bottom terrain following levels. (Note it should be inferior to hmin in case of Vtransform =1).

Then we have, with N the number of vertical levels:

- with the old transformation :

$$Cs(\sigma) = (1-\theta_b)\frac{sinh(\theta_s\ \sigma)}{sinh(\theta_s)} + \theta_b\left[\frac{0.5\ tanh\left((\sigma+0.5)\,\theta_s\right)}{tanh(0.5\ \theta_s)} - 0.5\right]$$

- with NEW_S_COORD defined :

$$sc = \frac{\sigma - N}{N} \tag{6.5}$$

$$csf = \frac{1. - cosh(\theta_s\ sc)}{cosh(\theta_s) - 1.} \quad \text{if } \theta_b > 0, \quad csf = -sc^2 \quad \text{otherwise} \tag{6.6}$$

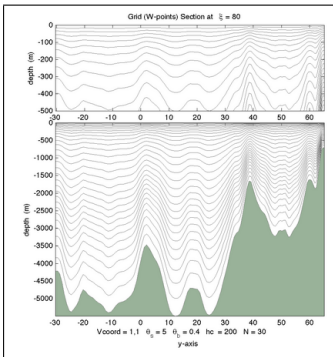
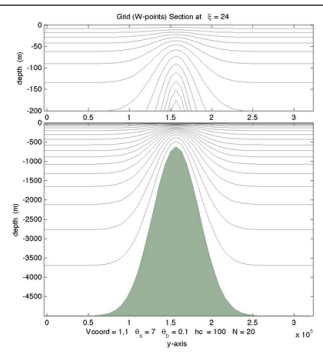
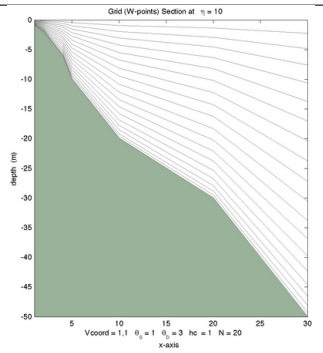
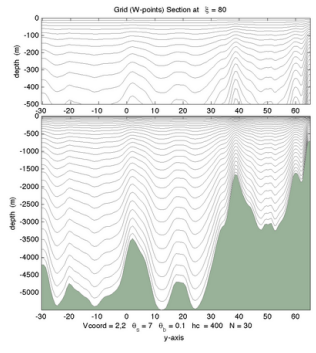
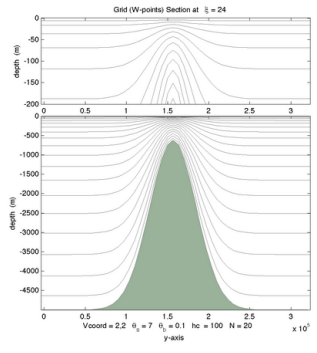
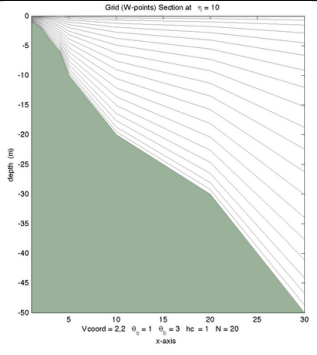
$$Cs(\sigma) = \frac{e^{\theta_b csf} - 1.}{1. - e^{-\theta_b}} \quad \text{if } \theta_s > 0, \quad Cs(\sigma) = csf \quad \text{otherwise} \tag{6.7}$$

Other parameters have to be set to prepare the grid file in `crocotools_param.m`:

vtransform	S-coordinate type (1: old- ; 2: new- coordinates). It is associated to #NEW_S_COORD cpp-keys in CROCO source code.
hmin	Minimum depth in meters. The model depth is cut a this level to prevent, for example, the occurrence of model grid cells without water. This does not influence the masking routines. At lower resolution, hmin should be quite large (for example 150m for dl=1/2). Otherwise, since topography smoothing is based on, the bottom slopes can be totally eroded.
hmax_coast	Maximum depth under the mask. It prevents selected isobaths (here 500 m) to go under the mask. If this is the case, this could be a source of problems for western boundary currents (for example).
hmax	Maximum depth
rtarget	This variable control the maximum value of the -parameter that measures the slope of the sigma layers (Beckmann and Haidvogel, 1993): To prevent horizontal pressure gradients errors, well in terrain-following coordinate models (Haney, 1991), realistic topography requires some smoothing. Empirical results have shown that reliable model results are obtained if does not exceed 0.2.
n_filter_deep_topo	Number of pass of a Hanning filter to prevent the occurrence of noise and isolated seamounts on deep regions.
n_filter_final	Number of pass of a Hanning filter at the end of the smoothing process to be sure that no noise is present in the topography.

The effects of theta_s, theta_b, hc, and N can be tested using the Matlab script : `croco_tools/Preprocessing_tools/test_vgrid.m`

Below are some examples of different vertical choices (Courtesy of ROMS-RUTGERS team) :

		
Vtransform=1, $\theta_S = 7.0, \theta_B = 0.1$	Vtransform=1, $\theta_S = 7.0, \theta_B = 1.0$	Vtransform=1, $\theta_S = 7.0, \theta_B = 3.0$
		
Vtransform =1, $\theta_S = 7.0, \theta_B = 1.0$	Vtransform=2, $\theta_S = 7.0, \theta_B = 1.0$	Vtransform=2, $\theta_S = 7.0, \theta_B = 3.0$

6.2 Wetting-Drying

The Wetting-Drying scheme is derived from John Warner's code (Rutgers ROMS) and adapted to the time stepping scheme of CROCO. The main idea is to cancel the outgoing momentum flux (not the incoming) from a grid cell if its total depth is below a threshold value (critical depth Dcrit between 5 and 20 cm according to local slope; Dcrit min and max adjustable in param.h). This scheme is tested in the Thacker case producing oscillations in a rotating bowl for which an analytical solution is known.

CHAPTER

SEVEN

NUMERICS

7.1 Overview

CROCO solves the primitive equations in an Earth-centered rotating environment. It is discretized in coastline- and terrain-following curvilinear coordinates using high-order numerical methods. It is a split-explicit, free-surface ocean model, where short time steps are used to advance the surface elevation and barotropic momentum, with a much larger time step used for temperature, salinity, and baroclinic momentum.

The complete time stepping algorithm is described in Shchepetkin and McWilliams (2005); see also Soufflet et al. (2016). The model has a 2-way time-averaging procedure for the barotropic mode, which satisfies the 3D continuity equation. The specially designed 3rd order predictor-corrector time step algorithm allows a substantial increase in the permissible time-step size.

Combined with the 3rd order time-stepping, a 3rd- or 5th-order, upstream-biased horizontal advection scheme (alternatively WENO or TVD for monotonicity preservation) allows the generation of steep gradients, enhancing the effective resolution of the solution for a given grid size (Shchepetkin and McWilliams, 1998; Soufflet et al., 2016; Menesguen et al., 2018, Borges et al., 2008). Because of the implicit diffusion in upstream advection schemes, explicit lateral viscosity is not needed in CROCO for damping numerical dispersion errors.

For vertical advection, SPLINE or WENO5 schemes are proposed (besides lower-order schemes). For SPLINES (default), an option for an adaptive, Courant-number-dependent implicit scheme is propose that has the advantage to render vertical advection unconditionally stable while maintaining good accuracy in locations with small Courant numbers (Shchepetkin, 2015). This is also available for tracers.

Tracers are treated similarly to momentum. A 3rd- or 5th-order upstream-biased horizontal advection scheme is implemented, but in regional configurations the diffusion part of this scheme is rotated along isopycnal surfaces to avoid spurious diapycnal mixing and loss of water masses (Marchesiello et al., 2009; Lemarié et al., 2012). For regional/coastal applications, a highly accurate pressure gradient scheme (Shchepetkin and McWilliams, 2003) limits the other type of errors (besides spurious diacpynal mixing) frequently associated with terrain-following coordinate models.

If a lateral boundary faces the open ocean, an active, implicit, upstream biased, radiation condition connects the model solution to the surroundings (Marchesiello et al., 2001). It comes with sponge layers for a better transition between interior and boundary solutions (explicit Laplacian diffusion and/or newtonian damping)

For nearshore problems, where waves becomes the dominant forcing of circulation, a vortex-force formalism for the interaction of surface gravity waves and currents is implemented in CROCO (Uchiyama et al., 2010).

CROCO can be used either as a Boussinesq/hystrostatic model, or a non-hydrostatic/non-Boussinesq model (NBQ; Auclair et al., 2018). The NBQ solver is relevant in problems from a few tens of meters to LES or DNS resolutions. It comes with shock-capturing advection schemes (WENO5, TVD) and fully 3D turbulent closure schemes (GLS, Smagorinsky).

CROCO includes a variety of additional features, e.g., 1D turbulent closure schemes (KPP, GLS) for surface and benthic boundary layers and interior mixing; wetting and drying; sediment and biological models; AGRIF interface for 2-way nesting; OASIS coupler for ocean-waves-atmosphere coupling…

7.2 Time Stepping

CROCO is discretized in time using a third-order predictor-corrector scheme (referred to as LFAM3) for tracers and baroclinic momentum. It is a split-explicit, free-surface ocean model, where short time steps are used to advance the surface elevation and barotropic momentum, with a much larger time step used for tracers, and baroclinic momentum. The model has a 2-way time-averaging procedure for the barotropic mode, which satisfies the 3D continuity equation. The specially designed 3rd order predictor-corrector time step algorithm is described in Shchepetkin and McWilliams (2005) and is summarized in this subsection.

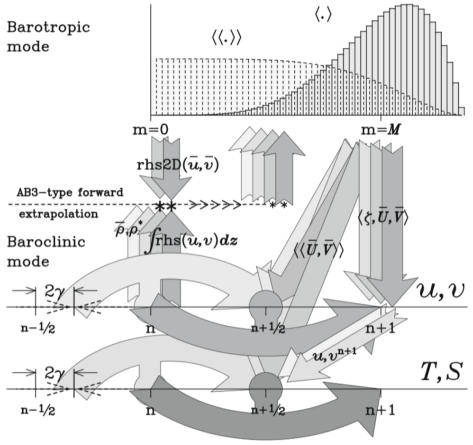


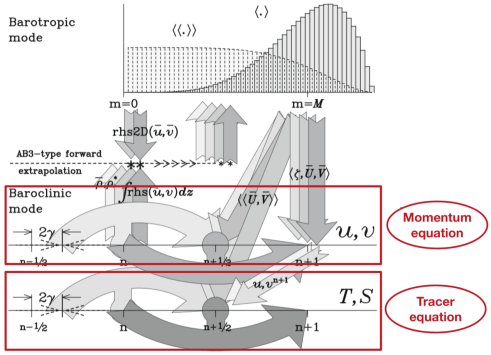
Fig. 1: Fig: schematic view of the Croco predictor-corrector hydrostatic kernel

General structure of the time-stepping:

```
call prestep3D_thread()     ! Predictor step for 3D momentum and tracers
call step2d_thread()        ! Barotropic mode
call step3D_uv_thread()    ! Corrector step for momentum
call step3D_t_thread()     ! Corrector step for tracers
```

7.2.1 3D momentum and tracers

Predictor-corrector approach : **Leapfrog (LF) predictor with 3rd-order Adams-Moulton (AM) interpolation (LFAM3 timestepping)**. This scheme is used to integrate 3D advection, the pressure gradient term, the continuity equation and the Coriolis term which are all contained in the RHS operator.



For a given quantity q

$$\begin{cases} q^{n+1,\star} &= q^{n-1} + 2\Delta t\ \mathrm{RHS}\left\{q^{n}\right\} & \text{(LF)} \\ q^{n+\frac{1}{2}} &= \dfrac{5}{12}\, q^{n+1,\star} + \dfrac{2}{3}\, q^{n} - \dfrac{1}{12}\, q^{n-1} & \text{(AM3)} \\ q^{n+1} &= q^{n} + \Delta t\ \mathrm{RHS}\left\{q^{n+\frac{1}{2}}\right\} & \text{(corrector)} \end{cases}$$

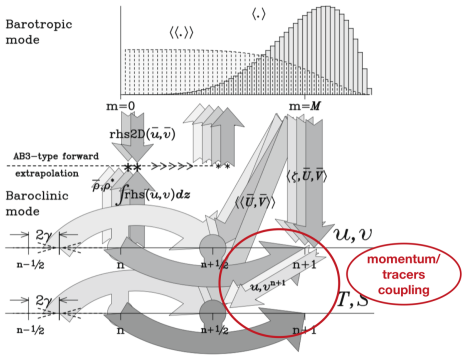
which can be rewritten in a compact way as used in the Croco code :

$$\begin{aligned} q^{n+\frac{1}{2}} &= \left(\frac{1}{2}-\gamma\right) q^{n-1} + \left(\frac{1}{2}+\gamma\right) q^{n} + (1-\gamma)\Delta t \ \mathrm{RHS}\left\{q^{n}\right\} \\ q^{n+1} &= q^{n} + \Delta t \ \mathrm{RHS}\left\{q^{n+\frac{1}{2}}\right\} \end{aligned}$$

with $\gamma = \frac{1}{6}$.

Physical parameterizations for vertical mixing, rotated diffusion and viscous/diffusion terms are computed once per time-step using an Euler step.

7.2.2 Tracers-momentum coupling



The numerical integration of internal waves can be studied using the following subsystem of equations

$$\left\{ \begin{array}{rcl} \partial_z w + \partial_x u &=& 0 \\ \partial_z p + \rho g &=& 0 \\ \partial_t u + \dfrac{1}{\rho_0}\partial_x p &=& 0 \\ \partial_t \rho + \partial_z (w\rho) &=& 0 \end{array} \right.$$

Predictor step:

$$\begin{aligned} \partial_x p^n = g\ \partial_x \left(\int_z^0 \rho^n dz \right) \quad &\rightarrow \quad u^{n+\frac{1}{2}} = \left(\frac{1}{2}-\gamma\right) u^{n-1} + \left(\frac{1}{2}+\gamma\right) u^{n} + (1-\gamma)\frac{\Delta t}{\rho_0}\left(\partial_x p^n\right) \\ w^n = -\int_{-H}^{z} \partial_x u^n dz' \quad &\rightarrow \quad \rho^{n+\frac{1}{2}} = \left(\frac{1}{2}-\gamma\right) \rho^{n-1} + \left(\frac{1}{2}+\gamma\right) \rho^{n} + (1-\gamma)\Delta t\ \partial_z (w^n \rho^n) \end{aligned}$$

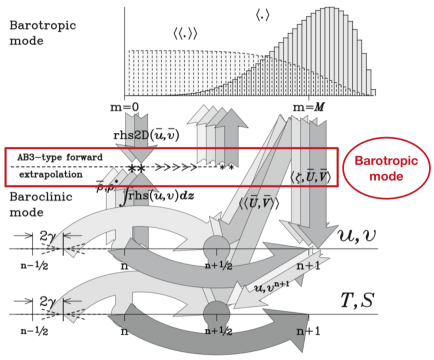
Corrector step:

$$\begin{aligned} \partial_x p^{n+\frac{1}{2}} = g\ \partial_x \left(\int_z^0 \rho^{n+\frac{1}{2}} dz \right) \quad &\rightarrow \quad u^{n+1} = u^n + \frac{\Delta t}{\rho_0}\left(\partial_x p^{n+\frac{1}{2}}\right) \\ w^{n+\frac{1}{2}} = -\int_{-H}^{z} \partial_x \left\{ \frac{3u^{n+\frac{1}{2}}}{4} + \frac{u^n + u^{n+1}}{8} \right\} dz' \quad &\rightarrow \quad \rho^{n+1} = \rho^n + \Delta t\ \partial_z (w^{n+\frac{1}{2}} \rho^{n+\frac{1}{2}}) \end{aligned}$$

Consequences:

- 3D-momentum integrated before the tracers in the corrector
- 2 evaluations of the pressure gradient per time-step
- 3 evaluations of the continuity equation per time-step

7.2.3 Barotropic mode



Generalized forward-backward (predictor-corrector)

1. AB3-type extrapolation

$$
\begin{aligned}
D^{m+\frac{1}{2}} &= H + \left(\frac{3}{2}+\beta\right)\zeta^{m} - \left(\frac{1}{2}+2\beta\right)\zeta^{m-1} + \beta\zeta^{m-2} \\
\overline{u}^{m+\frac{1}{2}} &= \left(\frac{3}{2}+\beta\right)\overline{u}^{m} - \left(\frac{1}{2}+2\beta\right)\overline{u}^{m-1} + \beta\overline{u}^{m-2}
\end{aligned}
$$

2. Integration of ζ

$$\zeta^{m+1} = \zeta^{m} - \Delta\tau\ \partial_x(D^{m+\frac{1}{2}}\overline{u}^{m+\frac{1}{2}})$$

3. AM4 interpolation

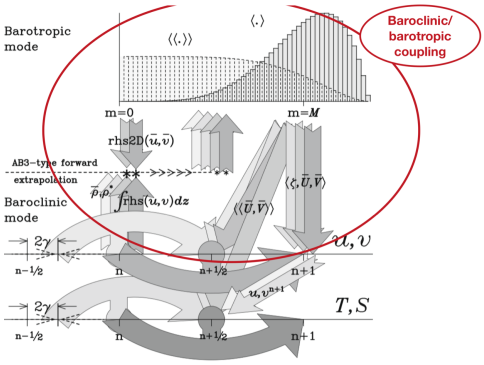
$$\zeta^{\star} = \left(\frac{1}{2}+\gamma+2\varepsilon\right)\zeta^{m+1} + \left(\frac{1}{2}-2\gamma-3\varepsilon\right)\zeta^{m} + \gamma\zeta^{m-1} + \varepsilon\zeta^{m-2}$$

4. Integration of $\bar{u}$

$$\overline{u}^{m+1} = \frac{1}{D^{m+1}}\left[D^{m}\overline{u}^{m} + \Delta\tau\ \mathrm{RHS2D}(D^{m+\frac{1}{2}}, \overline{u}^{m+\frac{1}{2}}, \zeta^{\star})\right]$$

where the parameter values are $(\beta, \gamma, \varepsilon) = (0.281105, 0.088, 0.013)$ except when the filter_none option is activated (see below).

7.2.4 Baroclinic-barotropic coupling

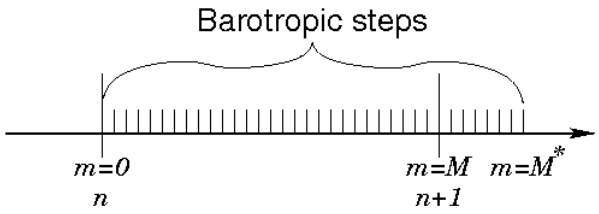


Slow forcing term of the barotrope by the barocline is extrapolated

$$\mathcal{F}_{3D}^{n+\frac{1}{2}} = \left\{\int \mathrm{rhs}(u,v)dz - \mathrm{rhs2D}(\bar{u},\bar{v})\right\}^{n+\frac{1}{2}} = \mathrm{Extrap}(\mathcal{F}_{3D}^{n}, \mathcal{F}_{3D}^{n-1}, \mathcal{F}_{3D}^{n-2})$$

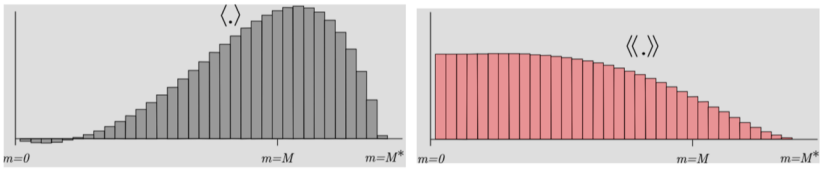
M2_FILTER_POWER option

Barotropic integration from n to $n + M^*\Delta\tau$ ($M^* \leq 1.5M$)



Because of predictor-corrector integration two barotropic filters are needed

- $\langle\zeta\rangle^{n+1} \rightarrow$ update of the vertical grid
- $\langle U\rangle^{n+1} \rightarrow$ correction of baroclinic velocities at time $n+1$
- $\langle\langle U\rangle\rangle^{n+\frac{1}{2}} \rightarrow$ correction of baroclinic velocities at time $n+\frac{1}{2}$



M2_FILTER_NONE option

Motivation: averaging filters can lead to excessive dissipation in the barotropic mode

Objective: put the minimum amount of dissipation to stabilize the splitting

Diffusion is introduced within the barotropic time-stepping rather than averaging filters by adapting the parameters in the generalized forward-backward scheme

$$(\beta, \gamma, \varepsilon) = (0.281105, 0.08344500 - 0.51358400\alpha_{\mathrm{d}}, 0.00976186 - 0.13451357\alpha_{\mathrm{d}})$$

with $\alpha_{\mathrm{d}} \approx 0.5$.

Remarks:

- This option may require to increase NDTFAST $= \Delta t_{3\mathrm{D}}/\Delta t_{2\mathrm{D}}$ because the stability constraint of the modified generalized forward-backward scheme is less than the one of the original generalized forward-backward scheme.
- The filter_none approach is systematically more efficient than averaging filters

7.2.5 Stability constraints

- **Barotropic mode** (note that considering an Arakawa C-grid divides the theoretical stability limit by a factor of 2)

$$\Delta t\sqrt{gH\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)} \leq 0.89$$

- **3D advection**

$$\alpha^x_{\mathrm{adv}} + \alpha^y_{\mathrm{adv}} + \beta\alpha^z_{\mathrm{adv}} \leq \alpha^\star_{\mathrm{horiz}}$$

where $\alpha^x_{\rm adv}$, $\alpha^y_{\rm adv}$, and $\alpha^z_{\rm adv}$ are the Courant numbers in each direction and $\beta = \alpha^\star_{\rm horiz}/\alpha^\star_{\rm vert}$ a coefficient arising from the fact that different advection schemes with different stability criteria may be used in the horizontal and vertical directions. Typical CFL values for $\alpha^\star_{\rm horiz}$ and $\alpha^\star_{\rm vert}$ with Croco time-stepping algorithm are

Advection scheme	Max Courant number ($\alpha^\star$)
C2	1.587
UP3	0.871
SPLINES	0.916
C4	1.15
UP5	0.89
C6	1.00

- **Internal waves**

$$\Delta t c_1 \sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}} \leq 0.843686$$

where c_1 the phase speed associated with the first (fastest) baroclinic mode.

- **Coriolis**

$$f \Delta t \leq 1.58$$

7.3 Advection Schemes

7.3.1 Lateral Momentum Advection

Related CPP options:

UV_HADV_UP3	Activate 3rd-order upstream biased advection scheme
UV_HADV_UP5	Activate 5th-order upstream biased advection scheme
UV_HADV_C2	Activate 2nd-order centred advection scheme (should be used with explicit momentum mixing)
UV_HADV_C4	Activate 4th-order centred advection scheme (should be used with explicit momentum mixing)
UV_HADV_C6	Activate 6th-order centred advection scheme (should be used with explicit momentum mixing)
UV_HADV_WENO5	Activate WENO 5th-order advection scheme
UV_HADV_TVD	Activate Total Variation Diminushing scheme

Preselected options:

```
# define UV_HADV_UP3
# undef  UV_HADV_UP5
# undef  UV_HADV_C2
# undef  UV_HADV_C4
# undef  UV_HADV_C6
# undef  UV_HADV_WENO5
# undef  UV_HADV_TVD
```

These options are set in set_global_definitions.h as the default UV_HADV_UP3 is the only one recommended for standard users.

7.3.2 Lateral Tracer advection

Related CPP options:

TS_HADV_UP3	3rd-order upstream biased advection scheme
TS_HADV_RSUP3	Split and rotated 3rd-order upstream biased advection scheme
TS_HADV_UP5	5th-order upstream biased advection scheme
TS_HADV_RSUP5	Split and rotated 5th-order upstream biased advection scheme with reduced dispersion/diffustion
TS_HADV_C4	4th-order centred advection scheme
TS_HADV_C6	Activate 6th-order centred advection scheme
TS_HADV_WENO5	5th-order WENOZ quasi-monotonic advection scheme for all tracers
BIO_HADV_WENO5	5th-order WENOZ quasi-monotone advection scheme for passive tracers (including biology and sediment tracers)

Preselected options:

```
# undef  TS_HADV_UP3
# define TS_HADV_RSUP3
# undef  TS_HADV_UP5
# undef  TS_HADV_RSUP5
# undef  TS_HADV_C4
# undef  TS_HADV_C6
# undef  TS_HADV_WENO5
# if defined PASSIVE_TRACER || defined BIOLOGY || defined SEDIMENT
#  define BIO_HADV_WENO5
# endif
```

TS_HADV_RSUP3 is recommended for realistic applications with variable bottom topography as it strongly reduces diapycnal mixing. It splits the UP3 scheme into 4th-order centered advection and rotated bilaplacian diffusion with grid-dependent diffusivity. It calls for CPP options in set_global_definitions.h for the explicit treatment of bilaplacian diffusion (see below). TS_HADV_RSUP3 is expensive in terms of computational cost and requires more than 30 sigma levels to perform properly. Therefore, for small domains dominated by open boundary fluxes, TS_HADV_UP5 may present a cheaper alternative and good compromise. TS_HADV_RSUP5 is still experimental but allows a decrease in numerical diffusivity compared to TS_HADV_RSUP3 by using 6th order rather than 4th-order centered advection (it resembles in spirit a split-rotated UP5 scheme but the use of bilaplacian rather than trilaplacian diffusion keeps it 3rd order). TS_HADV_C4 has no implicit diffusion and is thus accompanied by rotated Smagorinsky diffusion defined in set_global_definitions.h; it is not recommended for usual applications. For RSUP family, by default the diffusive part is oriented along geopotential.

7.3.3 Vertical Momentum advection

Related CPP options:

UV_VADV_SPLINES	4th-order compact advection scheme
UV_VADV_C2	2nd-order centered advection scheme
UV_VADV_WENO5	5th-order WENOZ quasi-monotone advection scheme
UV_VADV_TVD	Total Variation Diminushing (TVD) scheme

Preselected options:

```
#ifdef UV_VADV_SPLINES
#elif defined UV_VADV_WENO5
#elif defined UV_VADV_C2
#elif defined UV_VADV_TVD
#else
# define UV_VADV_SPLINES
# undef  UV_VADV_WENO5
# undef  UV_VADV_C2
# undef  UV_VADV_TVD
#endif
```

7.3.4 Vertical Tracer advection

Related CPP options:

TS_VADV_SPLINES	4th-order compact advection scheme
TS_VADV_AKIMA	4th-order centered advection scheme with harmonic averaging
TS_VADV_C2	2nd-order centered advection scheme
TS_VADV_WENO5	5th-order WENOZ quasi-monotone advection scheme

Preselected options:

```
#ifdef TS_VADV_SPLINES
#elif defined TS_VADV_AKIMA
#elif defined TS_VADV_WENO5
#elif defined TS_VADV_C2
#else
# undef  TS_VADV_SPLINES
# define TS_VADV_AKIMA
# undef  TS_VADV_WENO5
# undef  TS_VADV_C2
#endif
```

7.3.5 Adaptively implicit vertical advection

Related CPP options:

VADV_ADAPT_IMP	Activate adapative, Courant number dependent implicit advection scheme
VADV_ADAPT_PRED	Adaptive treatment at both predictor and corrector steps

Preselected options:

```
#ifdef VADV_ADAPT_IMP
# undef  VADV_ADAPT_PRED
# define UV_VADV_SPLINES
# undef  UV_VADV_C2
# undef  UV_VADV_WENO5
# undef  UV_VADV_TVD
#endif
#ifdef VADV_ADAPT_IMP
# define  TS_VADV_SPLINES
# undef   TS_VADV_AKIMA
# undef   TS_VADV_WENO5
# undef   TS_VADV_C2
#endif
```

7.3.6 Numerical details on advection schemes

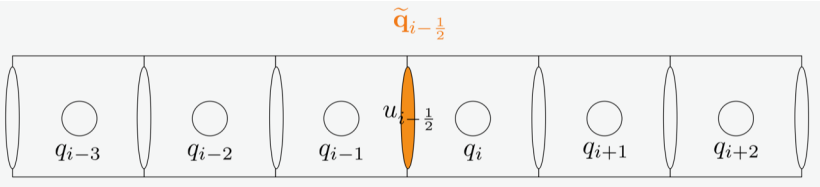


Fig. 2: Fig: variable location on an Arakawa C-grid. Tracer values are cell centered while velocities are defined on interfaces.

$$\partial_x(uq)|_{x=x_i} = \frac{1}{\Delta x_i}\left\{u_{i+\frac{1}{2}}\widetilde{q}_{i+\frac{1}{2}} - u_{i-\frac{1}{2}}\widetilde{q}_{i-\frac{1}{2}}\right\}$$

Linear advection schemes

$$\widetilde{q}^{\,\mathrm{C2}}_{i-\frac{1}{2}} = \frac{q_i + q_{i-1}}{2} \tag{7.1}$$

$$\widetilde{q}^{\,\mathrm{C4}}_{i-\frac{1}{2}} = \left(\frac{7}{6}\right)\widetilde{q}^{\,\mathrm{C2}}_{i-\frac{1}{2}} - \left(\frac{1}{12}\right)(q_{i+1} + q_{i-2}) \tag{7.2}$$

$$\widetilde{q}^{\,\mathrm{UP3}}_{i-\frac{1}{2}} = \widetilde{q}^{\,\mathrm{C4}}_{i-\frac{1}{2}} + \mathrm{sign}\left(\frac{1}{12}, u_{i-\frac{1}{2}}\right)(q_{i+1} - 3q_i + 3q_{i-1} - q_{i-2}) \tag{7.3}$$

$$\widetilde{q}^{\,\mathrm{C6}}_{i-\frac{1}{2}} = \left(\frac{8}{5}\right)\widetilde{q}^{\,\mathrm{C4}}_{i-\frac{1}{2}} - \left(\frac{19}{60}\right)\widetilde{q}^{\,\mathrm{C2}}_{i-\frac{1}{2}} + \left(\frac{1}{60}\right)(q_{i+2} + q_{i-3}) \tag{7.4}$$

$$\widetilde{q}^{\,\mathrm{UP5}}_{i-\frac{1}{2}} = \widetilde{q}^{\,\mathrm{C6}}_{i-\frac{1}{2}} - \mathrm{sign}\left(\frac{1}{60}, u_{i-\frac{1}{2}}\right)(q_{i+2} - 5q_{i+1} + 10q_i - 10q_{i-1} + 5q_{i-2} - q_{i-3}) \tag{7.5}$$

Split upwind schemes

Because odd-ordered advection schemes can be formulated as the sum of the next higher-order (centered) advection scheme with a dissipation term it is possible to split the purely centered and dissipative parts of UP3 and UP5 schemes. In this case the centered part is treated within the predictor-corrector framework while the flow-dependent dissipative part is treated with a one-step Euler scheme. Such splitting has two advantages:

1. It allows better stability for SUP3 and SUP5 schemes comapared to UP3 and UP5 schemes.
2. Isolating the dissipative part allows to rotate it in the neutral direction to reduce spurious diapycnal mixing (RSUP3 scheme).

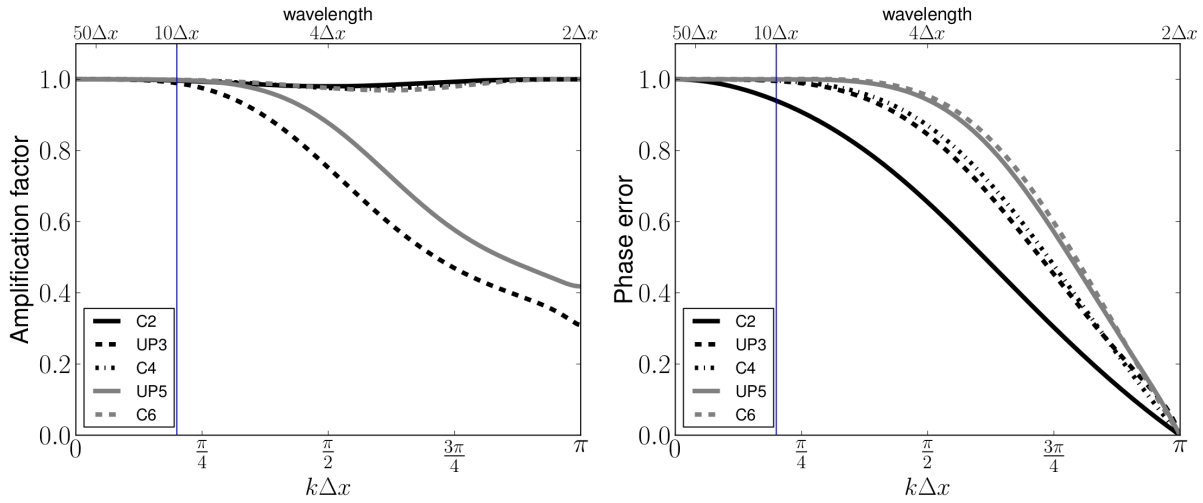


Fig. 3: Fig: amplification errors (left) and phase errors (right) for linear advection of order 2 to 6.

Splines reconstruction and Akima 4th-order schemes

Similar to a 4th-order compact scheme, the interfacial values for the splines reconstruction scheme are obtained as a solution of a tridiagonal problem

$$\mathrm{Hz}_{k+1}\widetilde{q}_{k-\frac{1}{2}} + 2(\mathrm{Hz}_k + \mathrm{Hz}_{k+1})\widetilde{q}_{k+\frac{1}{2}} + \mathrm{Hz}_k\widetilde{q}_{k+\frac{3}{2}} = 3(\mathrm{Hz}_k\overline{q}_{k+1} + \mathrm{Hz}_{k+1}\overline{q}_k)$$

where $\overline{q}_k$ values should be understood in a finite-volume sense (i.e. as an average over a control volume).

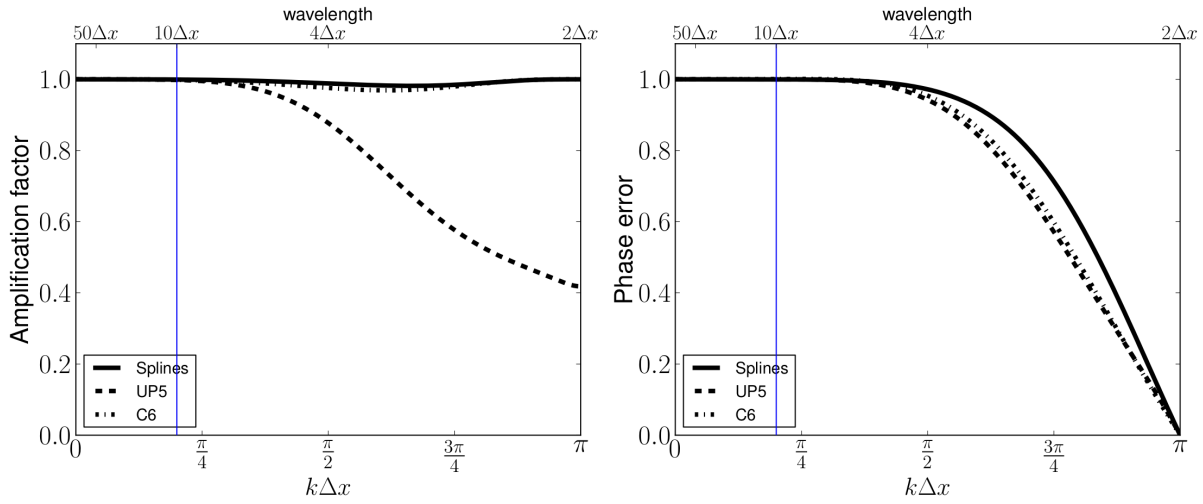


Fig. 4: Fig: amplification errors (left) and phase errors (right) for linear advection of order 5 and 6 and for Splines reconstruction.

The AKIMA scheme corresponds to a 4th-order accurate scheme where an harmonic averaging of the slopes is used instead of the algebraic average used for a standard C4 scheme

$$\widetilde{q}_{k+\frac{1}{2}} = \frac{q_{k+1}+q_k}{2} - \frac{\overline{\delta q}_{k+1} - \overline{\delta q}_k}{6} \qquad \overline{\delta q}_k = \begin{cases} 2\dfrac{\delta q_{k+\frac{1}{2}}\delta q_{k-\frac{1}{2}}}{\delta q_{k+\frac{1}{2}} + \delta q_{k-\frac{1}{2}}}, & \text{if } \delta q_{k+\frac{1}{2}}\delta q_{k-\frac{1}{2}} > 0 \\ 0, & \text{otherwise} \end{cases}$$

Adaptively implicit vertical advection

Idea: the vertical velocity Ω is split between an explicit and implicit contribution depending on the local Courant number

$$\Omega = \mathbf{\Omega^{(e)}} + \mathbf{\Omega^{(i)}}, \qquad \mathbf{\Omega^{(e)}} = \frac{\Omega}{f(\alpha^z_{\rm adv}, \alpha_{\rm max})}, \quad f(\alpha^z_{\rm adv}, \alpha_{\rm max}) = \begin{cases} 1, & \alpha^z_{\rm adv} \le \alpha_{\rm max} \\ \alpha/\alpha_{\rm max}, & \alpha^z_{\rm adv} > \alpha_{\rm max} \end{cases}$$

- $\Omega^{(e)}$ is integrated with an explicit scheme with CFL $\alpha_{\rm max}$.
- $\Omega^{(i)}$ is integrated with an implicit upwind Euler scheme.
- $f(\alpha^z_{\rm adv}, \alpha_{\rm max})$ is a function responsible for the splitting of Ω between an explicit and an implicit part.

This approach has the advantage to render vertical advection unconditionally stable and to maintain good accuracy in locations with small Courant numbers. The current implementation is based on the SPLINES scheme for the explicit part.

Total variation bounded scheme (WENO5)

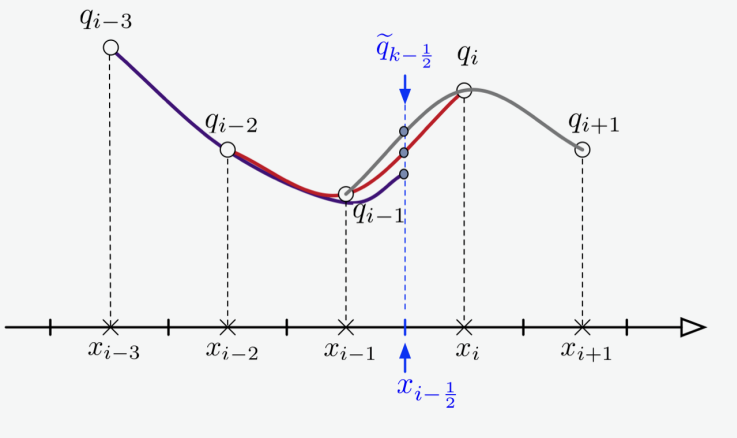


Fig. 5: Fig: different stencils used to evaluate the interfacial value $\widetilde{q}_{k+\frac{1}{2}}$ with WENO5 scheme

Nonlinear weighting between 3 evaluations of interfacial values based on 3 different stencils

$$\widetilde{q}_{k-\frac{1}{2}} = w_0 \widetilde{q}^{(0)}_{k-\frac{1}{2}} + w_1 \widetilde{q}^{(1)}_{k-\frac{1}{2}} + w_2 \widetilde{q}^{(2)}_{k-\frac{1}{2}}$$

where the weights are subject to the following constraints:

1. Convexity $\sum_{j=0}^{2} w_j = 1$.
2. ENO property (Essentially non oscillatory).
3. 5th-order if $q(x)$ is smooth.

The resulting scheme is not monotonicity-preserving but instead it is Total Variation Bounded (TVB).

Total variation diminishing scheme

Upwinding of nonlinear terms

In CROCO the nonlinear advection terms are formulated as in Lilly (1965) :

$$\partial_t(\mathrm{Hz}u) \quad + \quad \partial_x\left((\mathrm{Hz}\ u)u\right) + \partial_y\left((\mathrm{Hz}\ v)\ u\right) + \ \ldots \tag{7.6}$$

$$\partial_t(\mathrm{Hz}v) \quad + \quad \partial_x\left((\mathrm{Hz}\ u)v\right) + \partial_y\left((\mathrm{Hz}\ v)\ v\right) + \ \ldots \tag{7.7}$$

which are discretised with third order accuracy as

$$\left(\widetilde{(\mathrm{Hz}\ u)u}\right)_{i,j} = \widetilde{(\mathrm{Hz}\ u)}^{\mathrm{C4}}_{i,j}\ \widetilde{u}^{\mathrm{UP3}}_{i,j} \tag{7.8}$$

$$\left(\widetilde{(\mathrm{Hz}\ v)u}\right)_{i+\frac{1}{2},j+\frac{1}{2}} = \widetilde{(\mathrm{Hz}\ v)}^{\mathrm{C4}}_{i+\frac{1}{2},j+\frac{1}{2}}\ \widetilde{u}^{\mathrm{UP3}}_{i+\frac{1}{2},j+\frac{1}{2}} \tag{7.9}$$

where the direction for upwinding is selected considering

$$u^{\mathrm{upw}}_{i,j} = u_{i+\frac{1}{2},j} + u_{i-\frac{1}{2},j}, \qquad v^{\mathrm{upw}}_{i+\frac{1}{2},j+\frac{1}{2}} = (\mathrm{Hz}\ v)_{i,j+\frac{1}{2}} + (\mathrm{Hz}\ v)_{i+1,j+\frac{1}{2}}$$

7.4 Pressure gradient

This section is still under redaction. Meanwhile, please refer to Shchepetkin (2003)

Shchepetkin, A.F., McWilliams, J.C., 2003: A method for computing horizontal pressure-gradient force in an oceanic model with a non-aligned vertical coordinate. J. Geophys. Res. 108 (C3), 3090.

7.5 Equation of State

Related CPP options:

SALINITY	Activate salinity as an active tracer
NONLIN_EOS	Activate nonlinear equation of state
SPLIT_EOS	Activate the split of the nonlinear equation of state in adiabatic and compressible parts for reduction of pressure gradient errors

Preselected options:

```
# define SALINITY
# define NONLIN_EOS
# define SPLIT_EOS
```

The density is obtained from temperature and salinity (if SALINITY defined) via a choice of linear $\rho(T)$ or nonlinear $\rho(T,S,P)$ equation of state (EOS) described in Shchepetkin and McWilliams (2003). The nonlinear EOS corresponds to the UNESCO formulation as derived by Jackett and McDougall (1995) that computes in situ density as a function of potential temperature, salinity and pressure.

To reduce errors of pressure-gradient scheme associated with nonlinearity of compressibility effects, Shchepetkin and McWilliams (2003) introduced a Taylor expansion of this EOS that splits it into an adiabatic and a linearized compressible part (SPLIT_EOS):

$$\rho = \rho_0 + \rho_1(T,S) + q_1(T,S)\ |z|$$

where $\rho_1(T,S)$ is the sea-water density perturbation at the standard pressure of 1 Atm (sea surface), q_1 is the compressibility coefficient, and $|z|$ is absolute depth, i.e. the distance from free-surface to the point at which density is computed. This splitting of the EOS into two separate contributions allows for the representation of spatial derivatives of density as the sum of adiabatic derivatives and the compressible part. This makes it straightforward to remove pressure effects so as to reduce pressure gradient errors, compute neutral directions, enforce stable stratification, compute Brunt-Väisälä frequency etc.

The Brunt-Väisäla frequency N (at horizontal ρ and vertical w points) is defined by:

$$N^2 = -\frac{g}{\rho_0}\frac{\partial \rho_\theta}{\partial z}$$

where ρ_θ is potential density, .i.e., the density that a parcel would acquire if adiabatically brought to depth z_w.

Shchepetkin, A.F., McWilliams, J.C., 2003: A method for computing horizontal pressure-gradient force in an oceanic model with a non-aligned vertical coordinate. J. Geophys. Res. 108 (C3), 3090.

Shchepetkin, A.F., McWilliams, J.C., 2011. Accurate Boussinesq oceanic modeling with a practical, "stiffened" equation of state. Ocean Modell. 38, 41–70.

7.6 Wetting and Drying

The processes of wetting and drying have important physical and biological impacts on shallow water systems. Flooding and dewatering effects on coastal mud flats and beaches occur on various time scales ranging from storm surge, periodic rise and fall of the tide, to infragravity wave motions. To correctly simulate these physical processes with a numerical model requires the capability of the computational cells to become flooded and dewatered. Warner et al. (2013) proposed a method for wetting and drying based on an approach consistent with a cell-face blocking algorithm. The method allows water to always flow into any cell, but prevents outflow from a cell when the total depth in that cell is less than a user defined critical value. See Warner et al. (2013) for details.

The Wetting-Drying scheme is derived from John Warner's code (Rutgers ROMS) and adapted to the time stepping scheme of CROCO. The main idea is to cancel the outgoing momentum flux (not the incoming) from a grid cell if its total depth is below a threshold value (critical depth Dcrit between 5 and 20 cm according to local slope; Dcrit min and max adjustable in param.h). This scheme is tested in the Thacker case producing oscillations in a rotating bowl for which an analytical solution is known.

7.7 Non-Boussinesq Solver

CROCO can be used in a Boussinesq hystrostatic mode, or a non-hydrostatic, non-boussinesq mode (NBQ). The Non-Hydrostatic approach is based on the relaxation of the Boussinesq approximation instead of solving a Poisson system. It replaces the barotropic mode solver by a fully 3D fast mode solver, resolving all waves down to acoustic waves. The barotropic mode is part of the fast mode in this case. Depending on the physical problem, the sound speed can be decreased to the maximum wave velocity one wants to solve. The NH solver can be used in problems from a few tens of meters to LES or DNS resolutions. It comes with monotonicity preserving advection schemes (WENO5, TVD) and fully 3D turbulent closure schemes.

Related CPP options (for users):

NBQ	Activates Non-hysrostatic, non-Boussinesq solver

CHAPTER

EIGHT

PARAMETRIZATIONS

8.1 Vertical mixing parametrizations

CROCO contains a variety of methods for setting the vertical viscous and diffusive coefficients. The choices range from simply choosing fixed values to the KPP and the generic lengthscale (GLS) turbulence closure schemes. See Large (1998) for a review of surface ocean mixing schemes. Many schemes have a background molecular value which is used when the turbulent processes are assumed to be small (such as in the interior).

Related CPP options:

ANA_VMIX	Analytical definition
BVF_MIXING	Brunt-Vaisaleafrequency based
LMD_MIXING	K-profile parametrisation
GLS_MIXING	Generic lengthscale parametrisation

Preselected options:

```
NONE : default is no mixing scheme
```

8.1.1 Analytical definition

Related CPP options:

ANA_VMIX	Analytical definition

Preselected options:

```
NONE
```

A profile for mixing coeefficient $K_{m,s}(z)$ can be set in ana_vmix routine for variables Akv (viscosity) and Akt (diffusivity), which is called at each time step. In this case, background coeeficients read in croco.in can be used.

8.1.2 BVF mixing

Related CPP options:

BVF_MIXING	Brunt-Vaisala frequency based

Preselected options:

```
NONE
```

It computes diffusivity using a Brunt-Vaisala frequency based vertical mixing scheme. Viscosity is set to its background. In static unstable regime, diffusivity is enhanced.

- If $N^2(z) < 0$:

$$K_{m,s}(z) = 0.1 \text{ m}^2 \text{ s}^{-1}$$

- If $N^2(z) > 0$:

$$K_{m,s}(z) = 10^{-7}/\sqrt{N^2(z)}, \qquad K_{m,s}^{\min} \leq K_{m,s}(z) \leq K_{m,s}^{\max}$$

Default bounds are quite restrictive :

$$K_{m,s}^{\min} = 3 \times 10^{-5} \text{ m}^2 \text{ s}^{-1}, \qquad K_{m,s}^{\max} = 4 \times 10^{-4} \text{ m}^2 \text{ s}^{-1}$$

8.1.3 K-profile parametrization

Large, W., J. McWilliams, and S. Doney, Oceanic vertical mixing: A review and a model with nonlocal boundary layer parameterization, Rev. Geophys., 32, 363-403, 1994.

Related CPP options:

KPP-related options :

LMD_MIXING	K-profile parametrisation
LMD_SKPP	Activate surface boundary layer KPP mixing
LMD_SKPP2005	Activate surface boundary layer KPP mixing (2005 version)
LMD_BKPP	Activate bottom boundary layer KPP mixing
LMD_BKPP2005	Activate bottom boundary layer KPP mixing (2005 version)
LMD_RIMIX	Activate shear instability interior mixing
LMD_CONVEC	Activate convection interior mixing
LMD_DDMIX	Activate double diffusion interior mixing
LMD_NONLOCAL	Activate nonlocal transport for SKPP
LMD_LANGMUIR	Activate Langmuir turbulence mixing

Preselected options:

```
# ifdef LMD_MIXING
#  define LMD_SKPP
#  define LMD_BKPP
#  define LMD_RIMIX
#  define LMD_CONVEC
#  undef  LMD_DDMIX
#  define LMD_NONLOCAL
#  undef  LMD_LANGMUIR
# endif
```

#if defined LMD_SKPP # define LMD_SKPP2005 #endif #ifdef LMD_BKPP # undef LMD_BKPP2005 #endif

Surface boundary layer

- LMD_SKPP (Large et al, 1994)
  - Step 1 : Compute boundary layer depth $h_{bl}(z_r \rightarrow z_N)$

$$\mathrm{Ri}_b(z) = \frac{g(z_r - z)\left(\rho(z) - \rho_r\right)/\rho_0}{|\mathbf{u}(z) - (\mathbf{u}_h)_r|^2 + V_t^2(z)}, \qquad \mathrm{Ri}_b(-h_{bl}) = \mathrm{Ri}_{cr}$$

– Step 2 : In the stable case math::*(B_f > 0) : h_{bl} = min(h_{bl}, h_{ek}, h_{mo})*

$$h_{ek} = 0.7u_\star/f, \qquad h_{mo} = u_\star^3/(\kappa B_f).$$

– step 3 : Compute turbulent viscosity and diffusivity

$$K_{m,s}(z) = w_{m,s}\, h_{bl}\, G(z/h_{bl}), \qquad w_{m,s} = \kappa\, u_\star\, \psi_{m,s}(zB_f/u_\star^3)$$

Choice of the critical Richardson number Ri_{cr} : $\mathrm{Ri}_{cr} \in [0.15, 0.45]$

- LMD_SKPP2005 (Shchepetkin et al, 2005)
  - Criteria for h_{bl} : integral layer where production of turbulence by shear balances dissipation by the stratification

$$\mathrm{Cr}(z) = \int_z^\zeta \mathcal{K}(z') \left\{ |\partial_{z'}\mathbf{u}_h|^2 - \frac{N^2}{\mathrm{Ri}_{cr}} - C_{Ek}\, f^2 \right\} dz' + \frac{V_t^2(z)}{(\zeta - z)},\ \mathrm{Cr}(-h_{bl}) = 0$$

  - Consistent with the original KPP

$$\mathrm{Cr}(-h_{bl}) = 0 \Rightarrow \frac{(\zeta - z)\int_z^\zeta \mathcal{K}(z')N^2(z')dz'}{(\zeta - z)\int_z^\zeta \mathcal{K}(z') \left\{ |\partial_z\mathbf{u}_h|^2 - C_{Ek}\, f^2 \right\} dz' + V_t^2(z)} = \mathrm{Ri}_{cr}$$

Advantages :
-> consistent with Ekman problem
-> tends to give deeper boundary layers : $(\zeta - z)\int_z^\zeta |\partial_{z'}\mathbf{u}_h|^2\, dz' \geq |\mathbf{u}_h(z) - \mathbf{u}_h(\zeta)|^2$.

- cpp key LMD_LANGMUIR (McWilliams & Sullivan, 2000)

Following the work of McWilliams and Sullivan (2000), we introduce in KPP an enhancement factor E to the turbulent velocity scale as a function of the turbulent Langmuir number $La_t = \sqrt{u_\star/u_{Stokes}}$, but this function is taken as in Van Roekel et al. (2012) which gives good results in Li et al. (2016) – still assuming that Stokes drift is aligned with the surface wind stress:

$$w_{m,s} = \frac{\kappa\, u_\star}{\phi_{m,s}}\, E, \qquad E = \sqrt{1 + 0.104La_t^{-2} + 0.034La_t^{-4}}$$

Interior scheme

$$K_{m,s}(z) = K_{m,s}^{\mathrm{sh}}(z) + K_{m,s}^{\mathrm{iw}}(z) + K_{m,s}^{\mathrm{dd}}(z)$$

- cpp key LMD_RIMIX, RI_(H-V)SMOOTH (Large et al., 1994)

$$\mathrm{Ri}_g = N^2/\left[(\partial_z u)^2 + (\partial_z v)^2\right]$$

$$K_{m,s}^{\mathrm{sh}}(z) = \begin{cases} K_{0,c} & \mathrm{Ri}_g < 0 \quad \leftarrow [\mathrm{LMD\_CONVEC}] \\ K_0\left[1 - (\frac{\mathrm{Ri}_g}{\mathrm{Ri}_0})^3\right] & 0 < \mathrm{Ri}_g < \mathrm{Ri}_0 \\ 0 & \mathrm{Ri}_0 < \mathrm{Ri}_g \end{cases}$$

$$K_0 = 5 \times 10^{-3}\ \mathrm{m^2\ s^{-1}}, \mathrm{Ri}_0 = 0.7$$

- cpp key LMD_NUW_GARGETT (Gargett & Holloway)

$$K_m^{\rm iw}(z) = \frac{10^{-6}}{\sqrt{\max(N^2(z), 10^{-7})}}, \qquad K_s^{\rm iw}(z) = \frac{10^{-7}}{\sqrt{\max(N^2(z), 10^{-7})}}$$

- cpp key LMD_DDMIX (cf Large et al., 1994, eqns (31))

Bottom boundary layer

- cpp key LMD_BOTEK : Bottom Ekman layer

$$\begin{aligned}
h_{\rm Ek} &= \min\left\{\frac{0.3u_{\star,b}}{|f|}, h\right\} \\
\sigma_{k+\frac{1}{2}} &= (z_{k+\frac{1}{2}} - h)/h_{\rm Ek} \\
K^{\rm Ek}_{k+\frac{1}{2}} &= \max\left\{4\ \kappa\ u_{\star,b}\ h_{\rm Ek}\ \sigma(1-\sigma), K_{\rm min}\right\} \\
{\rm AKv}_{k+\frac{1}{2}} &= {\rm AKv}_{k+\frac{1}{2}} + K^{\rm Ek}_{k+\frac{1}{2}} \\
{\rm AKt}_{k+\frac{1}{2}} &= {\rm AKt}_{k+\frac{1}{2}} + K^{\rm Ek}_{k+\frac{1}{2}}
\end{aligned}$$

- cpp key LMD_BKPP (Bottom KPP 1994)

Same rationale than surface KPP but this time we search for the critical value $\mathrm{Ri}_{\rm cr}$ (≈ 0.3) starting from the bottom

$$h_{\rm bbl} = \min\left(h_{\rm bbl}, \frac{0.7u_{\star,b}}{|f|}\right) \qquad K_{m,s}(z) = \kappa\ u_{\star,b}\ h_{bbl}\ G(\sigma), \qquad \sigma = \frac{(z-h)}{h_{bbl}}$$

8.1.4 Generic length scale

GLS-related options :

GLS_MIXING	Activate Generic Length Scale scheme, default is k-epsilon (see below)
GLS_KOMEGA	Activate K-OMEGA (OMEGA=frequency of TKE dissipation) originating from Kolmogorov (1942)
GLS_KEPSILON	Activate K-EPSILON (EPSILON=TKE dissipation) as in Jones and Launder (1972)
GLS_GEN	Activate generic model of Umlauf and Burchard (2003)
CANUTO_A	Option for CANUTO A stability function (default, see below)
GibLau_78	Option for Gibson & Launder, 1978 stability function
MelYam_82	Option for Mellor & Yamada, 1982 stability function
KanCla_94	Option for Kantha & Clayson, 1994 stability function
Luyten_96	Option for Luyten, 1996 stability function
CANUTO_B	Option for CANUTO B stability function
Cheng_02	Option for Cheng, 2002 stability function

Preselected options for GLS:

```
#ifdef GLS_MIXING
# if   defined GLS_KOMEGA
# elif defined GLS_KEPSILON
# elif defined GLS_GEN
# else
#  define GLS_KEPSILON
# endif
```

(continues on next page)

(continued from previous page)

```
# if   defined CANUTO_A
# elif defined GibLau_78
# elif defined MelYam_82
# elif defined KanCla_94
# elif defined Luyten_96
# elif defined CANUTO_B
# elif defined Cheng_02
# else
#  define CANUTO_A
# endif
#endif
```

The objective of this section is to describe the current implementation of a Generic Length Scale (GLS) turbulence scheme in CROCO that computes the turbulent viscosity K_m and diffusivity K_s. First of all, as usually done in most implementations, the assumption of an horizontally homogeneous flow is made. Following Umlauf & Burchard (2003), the equations satisfied by the two prognostic variables k (the kinetic energy) and ψ (the generic length scale) are

$$\begin{aligned} \partial_t k &= \partial_z(K_k \partial_z k) + P + B - \varepsilon, & K_k &= K_m/\mathrm{Sc}_k \\ \partial_t \psi &= \partial_z(K_\psi \partial_z \psi) + \psi k^{-1}\left(\beta_1 P + \beta_3^{\pm} B - \beta_2 \varepsilon\right), & K_\psi &= K_m/\mathrm{Sc}_\psi \end{aligned}$$

where the β_j (j=1,3) are constants to be defined, P represents the TKE production by vertical shear $P = K_m\left[(\partial_z u)^2 + (\partial_z v)^2\right]$ and B the TKE destruction by stratification $B = -K_s N^2$ (with N^2 the local Brunt-Vaisala frequency). The dissipation rate ε is related to the generic length scale ψ following

$$\varepsilon = (c_\mu^0)^{3+p/n} k^{3/2+m/n} \psi^{-1/n}, \qquad \psi = (c_\mu^0)^p k^m l^n, \qquad l = (c_\mu^0)^3 k^{3/2} \varepsilon^{-1}$$

with l a mixing length and c_μ^0 a constant (whose value is between 0.526 and 0.555) to be defined. Depending on the parameter values for the triplet (m, n, p) the GLS scheme will either correspond to a $k-\varepsilon$, a $k-\omega$ or the so-called generic (Umlauf & Burchard, 2003) turbulence scheme (to simplify the code and because this scheme do not generally outperform other schemes, the possibility to use the so-called k-kl scheme is not implemented in Croco). Since the equations for e and ψ bear lots of similarities, to avoid excessive code duplication, a unique equation is solved for a quantity $\mathcal{T}_i$ encompassing k (when $i = i_{\mathrm{tke}}$) and ψ (when $i = i_{\mathrm{gls}}$, $i_{\mathrm{gls}} = i_{\mathrm{tke}} + 1$) such that

$$\partial_t \mathcal{T}_i = \partial_z(K_{\mathcal{T}_i} \partial_z \mathcal{T}_i) + (c_i^1 P + c_i^{3,\pm} B - c_i^2 \varepsilon), \qquad K_{\mathcal{T}_i} = K_m/\mathrm{Sc}_{\mathcal{T}_i}$$

where

$$\mathrm{Sc}_{\mathcal{T}_{i_{\mathrm{tke}}}} = \mathrm{Sc}_k, \qquad \mathrm{Sc}_{\mathcal{T}_{i_{\mathrm{gls}}}} = \mathrm{Sc}_\psi$$

and

$$\begin{aligned} c_i^1 &= (i_{\mathrm{gls}} - i) + (i - i_{\mathrm{tke}})\beta_1 e^{-1}\psi \\ c_i^2 &= (i_{\mathrm{gls}} - i) + (i - i_{\mathrm{tke}})\beta_2 e^{-1}\psi \\ c_i^{3,\pm} &= (i_{\mathrm{gls}} - i) + (i - i_{\mathrm{tke}})\beta_3^{\pm} e^{-1}\psi \end{aligned}$$

In practice this explains why in the code the two prognostic quantities k and ψ are stored in a single array $\mathrm{trb}(\mathrm{i}, \mathrm{j}, \mathrm{k}, \mathrm{ntime}, \mathrm{ngls})$ avec $\mathrm{ngls} = 2$, $i_{\mathrm{tke}} = 1$ and $i_{\mathrm{gls}} = 2$. Once the quantities k and ψ (hence ε) are known, the turbulent viscosity/diffusivity are given by

$$K_m = c_\mu \left(\frac{k^2}{\varepsilon}\right) = \frac{c_\mu}{(c_\mu^0)^3}(l\sqrt{k}), \qquad K_s = c_\mu^{'} \left(\frac{k^2}{\varepsilon}\right) = \frac{c_\mu^{'}}{(c_\mu^0)^3}(l\sqrt{k}).$$

where c_μ and $c_\mu^{'}$ are determined through so-called stability functions (see below).

Choice of parameter values and stability functions

A particular GLS occurence is defined by the following parameters :

- The exponents (m, n, p) in the definition of ε
- The Schmidt numbers Sc_k and Sc_ψ
- The coefficients β_j (j=1,3)
- The constant c_μ^0
- The stability functions which are generally function of

$$\alpha_M = \left(\frac{k}{\varepsilon}\right)^2 \left[(\partial_z u)^2 + (\partial_z v)^2\right], \qquad \alpha_N = \left(\frac{k}{\varepsilon}\right)^2 N^2$$

Where (m, n, p), Sc_k, Sc_ψ, β_j (j=1,3) are tied to a particular choice of GLS scheme (see table below) while c_μ^0, c_μ and $c_\mu^{'}$ are tied to a particular choice of stability function. The formulation of numerous stability functions can be reconciled when written using the generic form

$$\begin{aligned} c_\mu &= \frac{n_0 + n_1\alpha_N + n_2\alpha_M}{d_0 + d_1\alpha_N + d_2\alpha_M + d_3\alpha_N\alpha_M + d_4\alpha_N^2 + d_5\alpha_M^2} \\ c_\mu^{'} &= \frac{n_0^{'} + n_1^{'}\alpha_N + n_2^{'}\alpha_M}{d_0 + d_1\alpha_N + d_2\alpha_M + d_3\alpha_N\alpha_M + d_4\alpha_N^2 + d_5\alpha_M^2} \end{aligned}$$

where a given choice of stability function will define the parameter values for n_i, d_j, and $n_k^{'}$. In Croco, 7 options are available, these are referred to CANUTO-A, CANUTO-B, Gibson & Launder (1978), Mellor & Yamada (1982), Kantha & Clayson (1994), Luyten (1996), Cheng (2002).

Table 1: Table: parameter values corresponding to each particular GLS model

GLS model	m	n	p	β_1	β_2	β_3^-	β_3^+	Sc_e	Sc_ψ
$k-\omega$	0.5	-1	-1	0.555	0.833	-0.6	1	0.5	0.5
$k-\varepsilon$	1.5	-1	3	1.44	1.92	-0.4	1	1	0.7692
Gen	1	-0.67	0	1	1.22	0.05	1	1.25	0.9345

The quantities α_N and α_M in the formulation of c_μ and $c_\mu^{'}$ must satisfy some constraints to guarantee the regularity of numerical solutions. In CROCO, the following steps are done:

1. Apply the Galperin (1988) limitation i.e. $l \leq l_{\lim} = \beta_{\mathrm{galp}}\sqrt{2k/N^2}$ on ψ with $\beta_{\mathrm{galp}} = 0.53$. The first step is to use this mixing length $l_{\lim}$ to compute $\psi_{\min} = (c_\mu^0)^p k^m (l_{\lim})^n$ and to correct ψ to satisfy the constraint

$$\psi = \max(\psi, \psi_{\min})$$

here the max function is used since the exponent n is negative whatever the GLS scheme.

2. Compute the dissipation rate $\varepsilon = (c_\mu^0)^{3+p/n} k^{3/2+m/n} \psi^{-1/n}$ and correct it

$$\varepsilon = \max(\varepsilon, \varepsilon_{\min}), \qquad \varepsilon_{\min} = 10^{-12}\ \mathrm{m^2\ s^{-3}}$$

3. Compute α_N and α_M, and apply "stability and realisability" constraints following Umlauf & Burchard (2003) (their Sec. 4). A first constraint applies on α_N to ensure that $-\partial_{\alpha_N}(c_\mu^{'}/\alpha_N) > 0$ to prevent the occurence of oscillations in $c_\mu^{'}$. This translates into the following limiter

$$\alpha_N^{\min} = \frac{-(d_1 + n_0^{'}) + \sqrt{(d_1 + n_0^{'})^2 - 4d_0(d_4 + n_1^{'})}}{2(d_4 + n_1^{'})}, \qquad \alpha_N = \min\left(\max(0.73\alpha_N^{\min}), 10^{10}\right)$$

where the coefficient 0.73 is used to ensure the so-called realisability and has been empirically computed thanks to Table 3 in Umlauf & Burchard (2003) in order to satisfy their constraint (48). Then an upper limit is applied on α_M to ensure that $\partial_{\alpha_M}(c_\mu\sqrt{\alpha_M}) \geq 0$ which is also a prerequisite for stability reasons

$$\alpha_M^{\max} = \frac{d_0 n_0 + (d_0 n_1 + d_1 n_0)\alpha_N + (d_1 n_1 + d_4 n_0)\alpha_N^2 + d_4 n_1 \alpha_N^3}{d_2 n_0 + (d_2 n_1 + d_3 n_0)\alpha_N + (d_3 n_1)\alpha_N^2}, \qquad \alpha_M = \min(\alpha_M, \alpha_M^{\max})$$

Once those quantities are computed, stability functions are evaluated as well as the turbulent viscosity/diffusivity.

Surface and bottom boundary conditions

In current version of Croco, both k and ψ are formulated with Neumann boundary conditions at the top and at the bottom. However the nature of those boundary conditions also requires the determination of bottom and surface values for k and ψ.

- For turbulent kinetic energy, the "diagnostic" surface and bottom values are given by

$$k_{\rm sfc} = (u_\star^s/c_\mu^0)^2, \qquad k_{\rm bot} = (u_\star^b/c_\mu^0)^2$$

and simple homogeneous Neumann boundary conditions are applied

$$K_k \partial_z k|_{\rm sfc} = 0, \qquad K_k \partial_z k|_{\rm bot} = 0$$

In practice, due to the placement of k and ψ on the computational grid, the Neumann boundary condition is not applied strictly at the surface (resp. at the bottom) but at $z = z_N$ (resp. $z = z_1$) whereas the surface (resp. bottom) is located at $z = z_{N+1/2}$ (resp. $z = z_{1/2}$) with N the number of vertical levels (i.e. the number of cells in the vertical).

- For the generic length scale, a roughness is defined as

$$z_{0,s} = \max\left\{10^{-2}\ \mathrm{m}, \frac{C_{\rm ch}}{g}(u_\star^s)^2\right\}, \qquad C_{\rm ch} = 1400$$

at the surface and

$$z_{0,b} = \max\left\{10^{-4}\ \mathrm{m}, \mathrm{Zob}\right\}$$

at the bottom with Zob a user defined roughness length (usually $\mathrm{Zob} = 10^{-2}$ m). Again, the boundary conditions are applied at the center of the shallowest and deepest grid cells and not at their interfaces which means that the relevant length scales are

$$L_{\rm sfc} = \kappa\left(\frac{\Delta z_N}{2} + z_{0,s}\right), \qquad L_{\rm bot} = \kappa\left(\frac{\Delta z_1}{2} + z_{0,b}\right)$$

with κ the von Karman constant. Moreover TKE values are interpolated at $z = z_N$ and $z = z_1$

$$\widetilde{k}_{\rm sfc} = \frac{1}{2}\left(k_{\rm sfc} + k_{\rm N-1/2}\right), \qquad \widetilde{k}_{\rm bot} = \frac{1}{2}\left(k_{\rm bot} + k_{3/2}\right)$$

where $k_{\rm sfc}$ and $k_{\rm bot}$ are the diagnostic values given above. The "diagnostic" surface and bottom values for ψ are thus given by

$$\psi_{\rm sfc} = (c_\mu^0)^p (L_{\rm sfc})^n (\widetilde{k}_{\rm sfc})^m, \qquad \psi_{\rm bot} = (c_\mu^0)^p (L_{\rm bot})^n (\widetilde{k}_{\rm bot})^m$$

Then the surface and bottom flux are defined as

$$\mathcal{F}_\psi^{\rm sfc} = K_\psi \partial_z \psi|_{\rm sfc} = -n(c_\mu^0)^{p+1}\frac{\kappa}{\mathrm{Sc}_\psi}(\widetilde{k}_{\rm sfc})^{m+1/2}(L_{\rm sfc})^n$$

$$\mathcal{F}_\psi^{\rm bot} = K_\psi \partial_z \psi|_{\rm bot} = -n(c_\mu^0)^{p+1}\frac{\kappa}{\mathrm{Sc}_\psi}(\widetilde{k}_{\rm bot})^{m+1/2}(L_{\rm bot})^n$$

which correspond to the Neumann boundary conditions applied in the code.

8.2 Horizontal diffusion

8.2.1 Lateral Momentum Mixing

Related CPP options:

UV_MIX_GEO	Activate mixing on geopotential (constant depth) surfaces
UV_MIX_S	Activate mixing on iso-sigma (constant sigma) surfaces
UV_VIS2	Activate Laplacian horizontal mixing of momentum
UV_VIS4	Activate Bilaplacian horizontal mixing of momentum
UV_VIS_SMAGO	Activate Smagorinsky parametrization of turbulent viscosity (only with UV_VIS2)
UV_VIS_SMAGO3D	Activate 3D Smagorinsky parametrization of turbulent viscosity

Preslected options:

```
# ifdef UV_VIS2
#  define UV_MIX_S
#  define UV_VIS_SMAGO
# endif

#ifdef UV_VIS_SMAGO
# define VIS_COEF_3D
#endif

# ifdef UV_MIX_S
# elif defined UV_MIX_GEO
# else
# define UV_MIX_S /* Default*/
# endif# undef  UV_HADV_TVD
```

Explicit lateral momentum mixing may be only useful when implicit dissipation in UV_HADV_UP3 is not large enough to account for subgrid-scale turbulence resulting from large shear currents (for example in the case of western boundary currents). In this case, Smagorinsky parametrization is recommended (define UV_VIS2 below).

8.2.2 Lateral Tracer Mixing

Related CPP options:

TS_MIX_ISO	Activate mixing along isopycnal (isoneutral) surfaces
TS_MIX_GEO	Activate mixing along geopotential surfaces
TS_MIX_S	Activate mixing along iso-sigma surfaces
TS_DIF2	Activate Laplacian horizontal mixing of tracer
TS_DIF4	Activate Bilaplacian horizontal mixing of tracer
TS_MIX_IMP	Activate stabilizing correction of rotated diffusion (used with TS_MIX_ISO and TS_MIX_GEO)

Preslected options:

```
#ifdef TS_HADV_RSUP3 /* Rotated-Split 3rd-order scheme is: */
# define TS_HADV_C4  /* 4th-order centered advection */
# define TS_DIF4 /* + Hyperdiffusion  */
# define TS_MIX_GEO /* rotated along geopotential surfaces */
```

(continues on next page)

(continued from previous page)

```
# define TS_MIX_IMP /* with Semi-Implicit Time-Stepping */
# define DIF_COEF_3D
#endif
```

These options are preselected in set_global_definitions.h for compliance with Advection options.

8.3 Bottom friction

Related CPP options:

LIMIT_BSTRESS	Bottom stress limitation forstability
BSTRESS_FAST	Bottom stress computed in step3d_fast
BBL	Bottom boundary layer parametrization

Specification in croco.in:

```
bottom_drag:     RDRG [m/s],  RDRG2,  Zob [m],  Cdb_min, Cdb_max
                3.0d-04       0.d-3    0.d-3     1.d-4    1.d-1
```

- General form for 3D equations (cf get_vbc.F) :
  - If $z_{0,b} \neq 0$ $\rightarrow$ quadratic friction with log-layer ($C_{d,\min} \leq C_d \leq C_{d,\max}$)

$$\boldsymbol{\tau}_b = C_d \|\mathbf{u}_{k=1}\| \mathbf{u}_{k=1}, \qquad C_d = \left(\frac{\kappa}{\ln\left([z_1 - H]/z_{0,b}\right)} \right)^2$$

  - If $r_{\rm drg2} > 0$ $\rightarrow$ quadratic friction with C_d = constant}

$$\boldsymbol{\tau}_b = r_{\rm drg2} \|\mathbf{u}_{k=1}\| \mathbf{u}_{k=1},$$

  - Otherwise $\rightarrow$ linear friction

$$\boldsymbol{\tau}_b = r_{\rm drg} \mathbf{u}_{k=1},$$

- In the barotropic mode (cf step2D.F) :

$$\boldsymbol{\tau}_b^{\rm 2d} = (r_{\rm drg} + r_{\rm drg2} \|\bar{\mathbf{u}}\|) \bar{\mathbf{u}}$$

to be continued here for BSTRESS_FAST and BBL ...

BBL parametrization is detailed in the *Sediment and Biology models* section of the Doc.

CHAPTER

NINE

PARALLELISATION

CROCO has been designed to be optimized on both shared and distributed memory parallel computer architectures. Parallelization is done by two dimensional sub-domains partitioning. Multiple sub-domains can be assigned to each processor in order to optimize the use of processor cache memory. This allow super-linear scaling when performance growth even faster than the number of CPUs.

Related CPP options:

OPENMP	Activate OpenMP parallelization protocol
MPI	Activate MPI parallelization protocol
MPI_NOLAND	No computation on land only CPUs (needs preprocessing)
AUTO_TILING	Compute the best decomposition for OpenMP
PARALLEL_FILES	Output one file per CPU
NC4_PAR	Use NetCDF4 capabilities
XIOS	Dedicated CPU for output (needs XIOS installed)

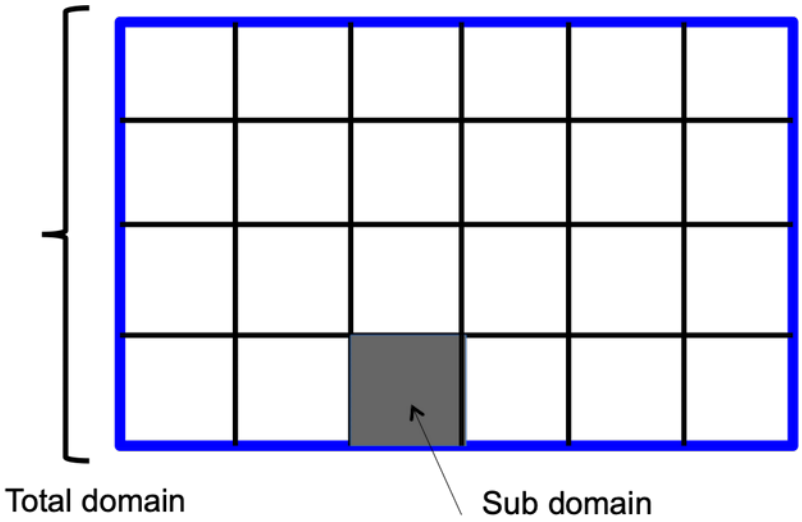
Preselected options:

```
# undef MPI
# undef OPENMP
# undef MPI_NOLAND
# undef AUTOTILING
# undef PARALLEL_FILES
# undef NC4_PAR
# undef XIOS
```

9.1 Parallel strategy overview

Two kind of parallelism are currently supported by CROCO : MPI (distributed memory) and OpenMP (shared memory). COROC doesn't currently support hybrid parallelisation : use of cpp keys MPI or OPENMP is exclusive.

9.1.1 OpenMP (#define OPENMP)



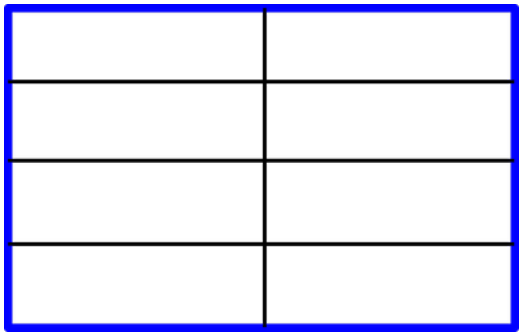
Variables in param.h:

- NPP : number of threads
- NSUB_X : number of tiles in XI direction
- NSUB_E : number of threads in ETA direction

NSUB_X x NSUB_E has to be a multiple of NPP. Most of the time, we set NPP=NSUB_X x NSUB_E

Example 1:

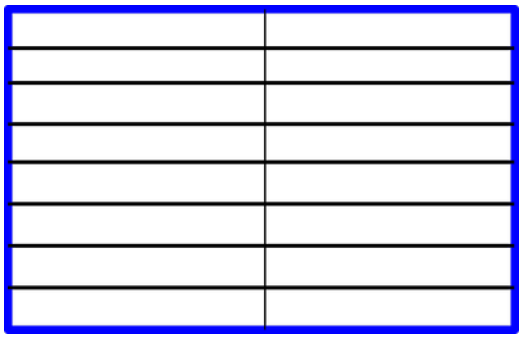
One node with 8 cores: NPP=8, NSUB_X=2, NSUB_ETA=4



Each thread computes **one** sub-domain.

Example 2:

Still one node with 8 cores: NPP=8, NSUB_X=2, NSUB_E=8



Each thread computes **two** sub-domains.

Code structure

- OpenMP is **NOT** implemented at loop level
- but uses a domain decomposition (similar to MPI) with parallel region
- use of *First touch initialisation* so working arrays are attached to the same thread
- working arrays have the size of the sub-domain only

```
C$OMP PARALLEL
    Call step3D_t_thread()
C$OMP END PARALLEL
```

Fig. 1: Example of a parallel region

```
    Do tile=my_first,my_last   ! Loop on the tiles computed
                               ! by the current thread
      Call compute_1(tile)   ! No Synchronisation needed
      Call compute_2(tile)   ! by these procedures
    Enddo
C$OMP BARRIER ! synchronisation

    Do tile=my_first,my_last   ! Loop on the tiles computed
                               ! by the current thread
      Call compute_3(tile)   !  No Synchronisation needed
      Call compute_4(tile)   !
    Enddo
C$OMP BARRIER ! synchronisation
```

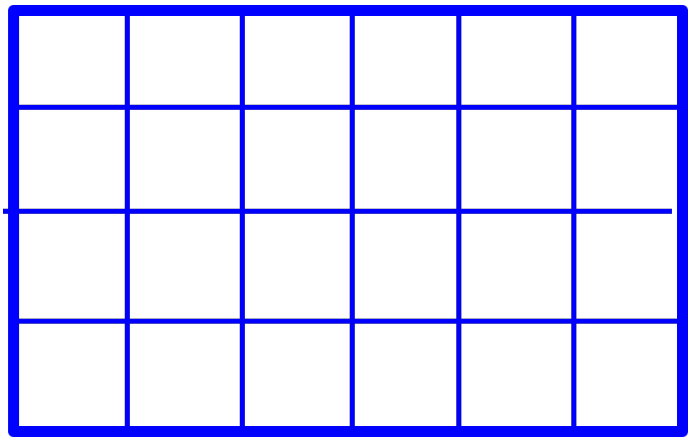
Fig. 2: Inside a parallel region

Here Compute_1 and Compute2 can't write on the same index of a global array.

9.1.2 MPI (#define MPI)

Variables in param.h:

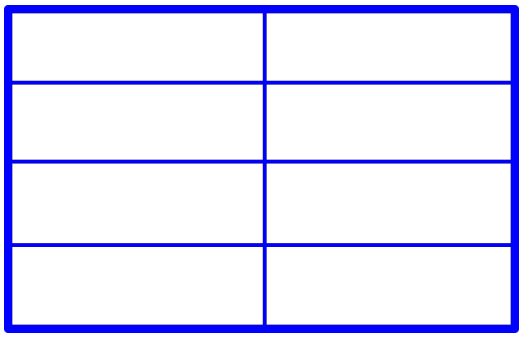
- NP_XI : decompostion in XI direction
- NP_ETA : decomposition in ETA direction
- NNODES : number of cores (=NP_XI x NP_ETA, except with MPI_NOLAND)
- NPP = 1
- NSUB_X and NSUB_ETA, number of sub-tiles (almost always =1)



Example 1:

8 cores:

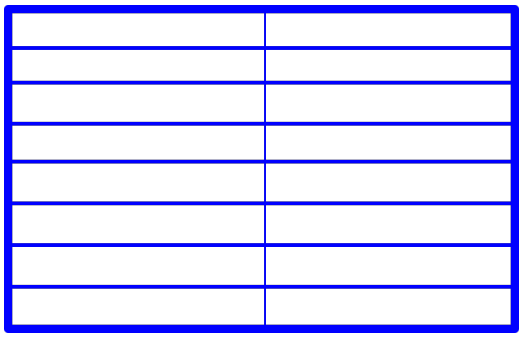
- NP_XI=2, NP_ETA=4, NNODES=8
- NPP=1, NSUB_X=1, NSUB_ETA=1



Example 2:

8 cores:

- NP_XI=2, NP_ETA=4, NNODES=8
- NPP=1, NSUB_X=1, NSUB_ETA=2

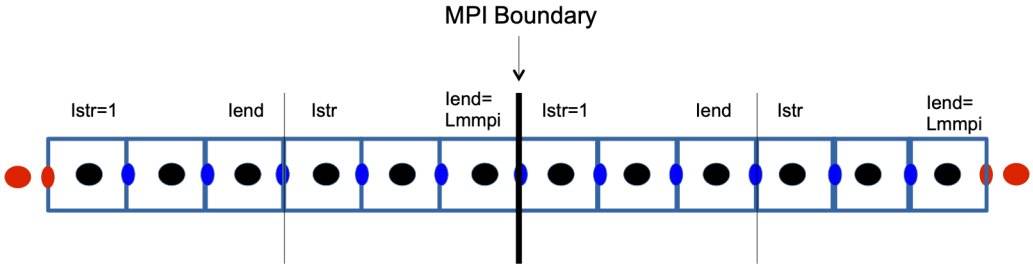


9.2 Loops and indexes

Parallel/sequential correspondance:

Decomposition:

Example : 2 MPI domains, with 2 sub-domains (OpenMP or not) by domain MPI



Istr, Iend are the limits of the sub-domains (without overlap). There are calculated dynamically.

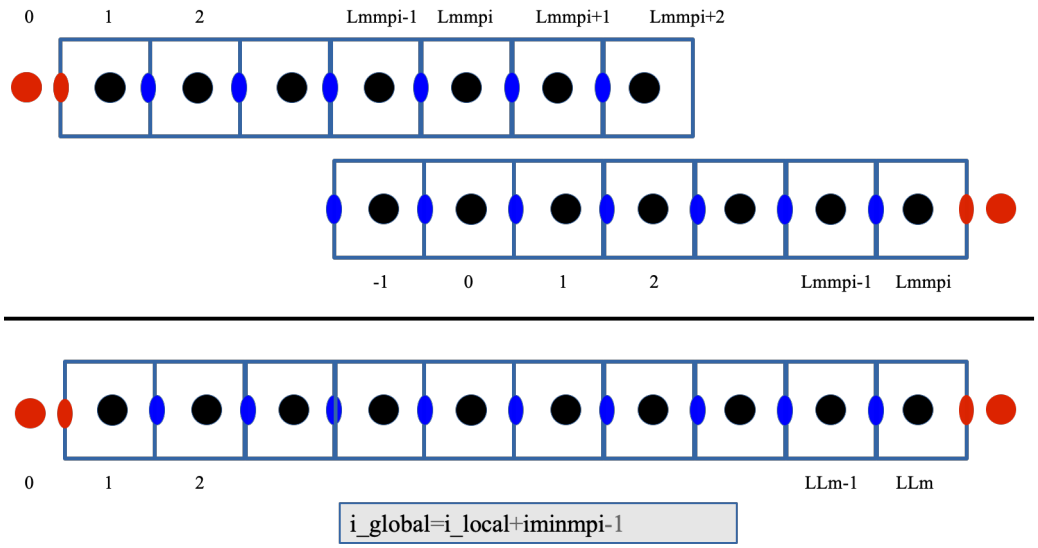


Fig. 3: Decomposition on 2 sub-domain (up), total (sequential) domain (bottom)

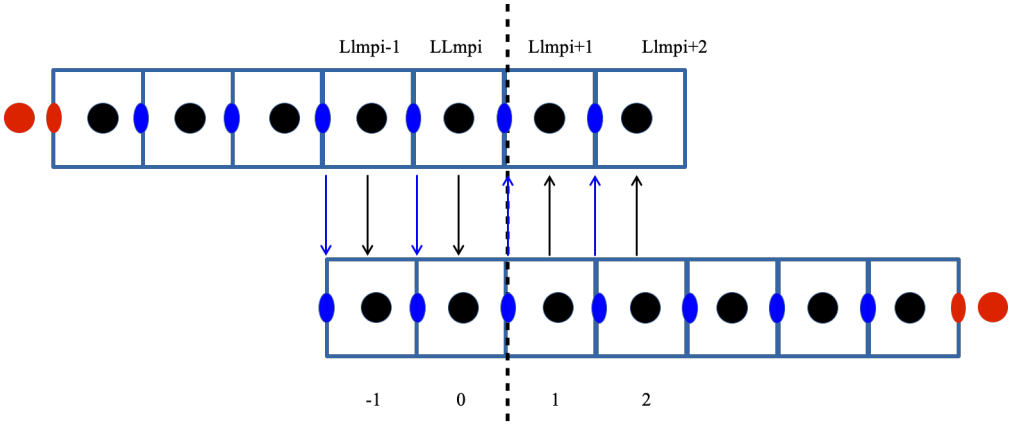
```
      Subroutine compute_1(tile)
      Integer tile, trd
      #include « private_scratch.h » ! Private work arrays
                                     ! A2d, A3d
      #include « compute_tile_bounds.h » Compute Istr, Iend

      trd=0
C$    trd=omp_get_thread_num()
      Call compute_1_tile(Istr,Iend,Jstr,Jend,A2d(1,1,trd),A3d(1,1,trd))
      Return
      end
```

Fig. 4: Computation of Istr, Iend and use of working arrays

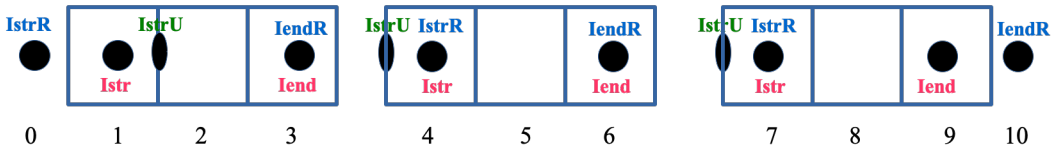
9.3 Exchanges

CROCO makes use 2 or 3 ghost cells depending on the numerical schemes chosen.



In the example above (2 ghosts cells), for correct exchanges, after computation:

- η has to be valid on (1:Iend)
- u has to be valid on (1:Iend) except on the left domain (2:Iend)



IstrU is the limit of validity at U point

```
Subroutine compute_1_tile(Istr, Iend …)
...
#include « compute_auxiliary_bounds » ! Compute IstrU, IstrR
…
```

Fig. 5: Compuation of auxiliary indexes

9.4 Dealing with outputs

By default, with MPI activated input and output files are treated in a pseudo-sequential way, and one NetCDFfile corresponds to the whole domain. This has drawbacks when using a large number of computational cores, since each core is writing its part of the domain sequentially, the time dedicated to outputs increase with the number of cores. Three alternatives are implemented within CROCO.

Splited files (#define PARALLEL_FILES)

In this case, each core is writing its part only of the domain in separated files (one per MPI domain). This writing is performed concurrently. One other advantage is to avoid the creation of huge output files. The domain related output files can be recombined using ncjoin utility (in fortran) compiled in the same time than CROCO. Note that in this case, input files have to be splited as well, using partit utility.

Parallel NetCDF(#define NC4_FILES)

This option requires NetcDF4 verion, installed with parallel capabilities. All cores are writing concurrently but in the same time.

IO server (#define XIOS)

XIOS is an external IO server interfaced with CROCO. Informations about use and installation can be found there https://forge.ipsl.jussieu.fr/ioserver. In this case, output variables are defined in .xml files. See also the Diagnostics chapter.

CHAPTER

TEN

ATMOSPHERIC SURFACE BOUNDARY LAYER

Related CPP options:

BULK_FLUX	Activate bulk formulation for surface turbulent fluxes (by default, COARE3p0 parametrizarion is used)
BULK_ECUMEV0	Use ECUMEv0 bulk formulation instead of COARE3p0 formulation
BULK_ECUMEV6	Use ECUMEv6 bulk formulation instead of COARE3p0 formulation
BULK_WASP	Use WASP bulk formulation instead of COARE3p0 formulation
BULK_GUSTINESS	Add in gustiness effect on surface wind module. Can be used for both bulk parametrizations.
BULK_LW	Add in long-wave radiation feedback from model SST
SFLUX_CFB	Activate current feedback on ... (Renault et al., 2020)
CFB_STRESS	... surface stress (used by default when SFLUX_CFB is defined)
CFB_WIND_TRA	... surface tracers (used by default when SFLUX_CFB is defined)
SST_SKIN	Activate skin sst computation (Zeng & Beljaars, 2005)
ONLINE	Read native files and perform online interpolation on CROCO grid (default cubic interpolation)
QCORRECTION	Activate heat flux correction around model SST (if BULK_FLUX is undefined)
SFLX_CORR	Activate freshwater flux correction around model SSS (if BULK_FLUX is undefined)
ANA_DIURNAL_SW	Activate analytical diurnal modulation of short wave radiations (only appropriate if there is no diurnal cycle in data)

By default COARE3p0 parametrization is used with GUSTINESS effects. To change bulk parametrization you have to define one the following cpp keys (not additional) :

- define BULK_ECUMEV0 to use ECUME_v0 parametrization
- define BULK_ECUMEV6 to use ECUME_v6 parametrization
- define BULK_WASP to use WASP parametrization

Warning : it is possible to add GUSTINESS effects for all parametrizations by defining BULK_GUSTINESS cpp key

ONLINE CPP options:

ONLINE option is an alternative to pre-processing of surface forcing data, that can be useful for long-term simulations, especially if handling multiple configurations. ONLINE option calls for CUBIC_INTERP in set_global_definitions.h.

ECMWF	Use ECMWF atm fluxes
AROME	Use METEO FRANCE fluxes
READ_PATM	Read atmospheric pressure instead of use default reference pressure and take into account atmospherical pressure gradient in the equations
OBC_PATM	In case of READ_PATM, inverse barometer effect to the open boundaries if atmospherical pressure is read in meteo file.

Preselected options (cppdefs.h):

```
# undef BULK_FLUX
# ifdef BULK_FLUX
#  undef  BULK_ECUMEV0
#  undef  BULK_ECUMEV6
#  undef  BULK_WASP
#  define BULK_GUSTINESS
#  define BULK_LW
#  undef  SST_SKIN
#  undef  ANA_DIURNAL_SW
#  undef  ONLINE
#  ifdef ONLINE
#   undef  AROME
#   undef  ERA_ECMWF
#  endif
#  undef READ_PATM
#  ifdef READ_PATM
#   define OBC_PATM
#  endif
# else
#  define QCORRECTION
#  define SFLX_CORR
#  undef  SFLX_CORR_COEF
#  define ANA_DIURNAL_SW
# endif
# undef  SFLUX_CFB
# undef  SEA_ICE_NOFLUX
```

Preselected options (cppdefs_dev.h):

```
#ifdef BULK_FLUX
# ifdef ONLINE
#  define CUBIC_INTERP
# endif
# ifdef BULK_ECUMEV0
#  define BULK_GUSTINESS
# elif defined BULK_ECUMEV6
#  define BULK_GUSTINESS
# elif defined BULK_WASP
#  define BULK_GUSTINESS
# endif
#endif
```

```
#ifdef SFLUX_CFB
# ifdef BULK_FLUX
#  define CFB_STRESS
#  define CFB_WIND_TRA
# else
#  undef  CFB_STRESS
#  undef  CFB_WIND_TRA
# endif
#endif
```

CHAPTER

ELEVEN

OPEN BOUNDARIES CONDITIONS

If a lateral boundary faces the open ocean, robust open boundary conditions (OBCs) are needed (Marchesiello et al., 2001). Forcing of tracer and baroclinic flow is applied via an adaptive radiation condition, which helps perturbations to leave the domain with only a small effect on the interior solution. The same method can be applied to the depth-averaged flow, but (for tidal forcing in particular) we generally prefer the incoming characteristic of the shallow water system as in Flather-type conditions (Marchesiello et al., 2001; Blayo and Debreu, 2005). This allows long-wave data to be forced in, while those generated inside the domain can leave it, which also guarantees the quasi-conservation of mass and energy across the open boundary. A Sponge layer is added near the open boundaries to limit small-scale effects and ease the transition between the interior solution and the boundary data. The boundary data can be applied only at the boundary (see BRY strategy below) or in a nudging layer (CLIMATOLOGY strategy).

This set of OBCs are given as default if no other choice is made. They have performed well in most applications, from the deep ocean to the coastal areas. For details, refer to Marchesiello et al. (2001) and Blayo and Debreu (2005):

- Marchesiello, P., J.C. McWilliams, and A. Shchepetkin, 2001: Open boundary conditions for long-term integration of regional oceanic models. Ocean Modelling, 3, 1-20.
- Blayo E. anb L. Debreu, 2005: Revisiting open boundary conditions from the point of view of characteristic variables. Ocean Modelling, 9, 231-252.

11.1 OBC

Related CPP options:

OBC_EAST	Open eastern boundary
OBC_WEST	Open western boundary
OBC_SOUTH	Open southern boundary
OBC_NORTH	Open northern boundary

Related CPP options:

OBC_M2SPECIFIED	Activate specified OBCs for barotropic velocities
OBC_M2CHARACT	Activate OBCs from characteristic methods for barotropic velocities (default)
OBC_M2ORLANSKI	Activate radiative OBCs for barotropic velocities
OBC_VOLCONS	Enforce mass conservation at open boundaries (with OBC_M2ORLANSKI)
OBC_M3SPECIFIED	Activate specified OBCs for baroclinic velocities
OBC_M3ORLANSKI	Activate radiative OBCs for baroclinic velocities (default)
OBC_TSPECIFIED	Activate specified OBCs for tracers
OBC_TORLANSKI	Activate radiative OBCs for tracers (default)
OBC_TUPWIND	Activate upwind OBCs for tracers

For non-tidal forcing, the combination of OBC_M2ORLANSKI and OBC_VOLCONS often provides the best performances in terms of transparency of barotropic flow at the open boundaries. However, OBC_M2CHARACT

is near as good and also provides the best conditions for tidal forcing. It is therefore set as default in cppdefs_dev.h.

Preselected options (in cppdefs_dev.h but set your own choice in cppdefs.h if needed):

```
# undef  OBC_M2SPECIFIED
# define OBC_M2CHARACT
# undef  OBC_M2ORLANSKI
# ifdef OBC_M2ORLANSKI
#  define OBC_VOLCONS
# endif
# define OBC_M3ORLANSKI
# define OBC_TORLANSKI
# undef  OBC_M3SPECIFIED
# undef  OBC_TSPECIFIED
```

11.2 Sponge Layer

SPONGE is preselected in cppdefs.h and calls for SPONGE_GRID in cppdefs_dev.h. SPONGE_GRID selects the sponge layer extension (10 points with cosine shape function) and viscosity and diffusivity values according to the horizontal resolution (limited by the CFL stability conditions).

Related CPP options:

SPONGE	Activate areas of enhanced viscosity and diffusivity near lateral open boundaries.
SPONGE_GRID	Automatic setting of the sponge width and value
SPONGE_DIF2	Sponge on tracers (default)
SPONGE_VIS2	Sponge on momentum (default)
SPONGE_SED	Sponge on sediment (default)

11.3 Nudging layers

The nudging layer has the same extension as the sponge layer. In nudging layers, tracer and momentum fields are nudged towards climatological values at a time scale Tau_out (possibly different for momentum and tracers) that is given in namelist croco.in

Related CPP options:

ZNUDGING	Activate nudging layer for sea level
M2NUDGING	Activate nudging layer for barotropic velocities
M3NUDGING	Activate nudging layer for baroclinic velocities
TNUDGING	Activate nudging layer for tracers
ROBUST_DIAG	Activate nudging over the whole domain

11.4 Lateral forcing

11.4.1 CLIMATOLOGY strategy

Related CPP options:

CLIMATOLOGY	Activate processing of 2D/3D data (climatological or simulation/reanalysis) used as forcing at the open boundary points + nudging layers
ZCLIMATOLOGY	Activate processing of sea level
M2CLIMATOLOGY	Activate processing of barotropic velocities
M3CLIMATOLOGY	Activate processing of baroclinic velocities
TCLIMATOLOGY	Activate processing of tracers

11.4.2 BRY strategy

FRC_BRY is useful for inter-annual forcing on high-resolution domains.^FRC_BRY is compatible with CLIMATOLOGY that can still be used for nudging layers.

Related CPP options:

FRC_BRY	Activate processing of 1D/2D data used as forcing at open boundary points strictly
Z_FRC_BRY	Activate open boundary forcing for sea level
M2_FRC_BRY	Activate open boundary forcing for barotropic velocities
M3_FRC_BRY	Activate open boundary forcing for baroclinic velocities
T_FRC_BRY	Activate open boundary forcing for tracers

Preselected options (cppdefs.h):

```
# define CLIMATOLOGY
# ifdef CLIMATOLOGY
#  define ZCLIMATOLOGY
#  define M2CLIMATOLOGY
#  define M3CLIMATOLOGY
#  define TCLIMATOLOGY
#  define ZNUDGING
#  define M2NUDGING
#  define M3NUDGING
#  define TNUDGING
#  undef ROBUST_DIAG
# endif
# undef FRC_BRY
# ifdef FRC_BRY
#  define Z_FRC_BRY
#  define M2_FRC_BRY
#  define M3_FRC_BRY
#  define T_FRC_BRY
# endif
```

CHAPTER

TWELVE

RIVERS

Related CPP options:

PSOURCE	Activate point sources (rivers)
ANA_PSOURCE	use analytical vertical profiles for point sources (set in set_global_definitions.h)
PSOURCE_NCFILE	Read variable river transports in netcdf file
PSOURCE_NCFILE_TS	Read variable river concentration in netcdf file

ANA_PSOURCE gives the vertical distribution of point source outflow. The default shape is an uniform distribution along the vertical. The vertical shape can be customized in subroutine *ana_psource* in *analytical.F*

An example of runoff file is given below

```
netcdf croco_runoff {
dimensions:
      qbar_time = 28193 ;
      n_qbar = 9 ;
      runoffname_StrLen = 30 ;
      two = 2 ;
      temp_src_time = 10690 ;
      salt_src_time = 10690 ;
variables:
      double qbar_time(qbar_time) ;
              qbar_time:long_name = "runoff time" ;
              qbar_time:units = "days" ;
              qbar_time:cycle_length = 0. ;
              qbar_time:long_units = "days since 1900-01-01" ;
      char runoff_name(n_qbar, runoffname_StrLen) ;
              runoff_name:long_name = "runoff name" ;
      double runoff_position(n_qbar, two) ;
              runoff_position:long_name = "position of the runoff (by line) in the␣
↪CROCO grid" ;
      double runoff_direction(n_qbar, two) ;
              runoff_direction:long_name = "direction/sense of the runoff (by␣
↪line) in the CROCO grid" ;
      double Qbar(n_qbar, qbar_time) ;
              Qbar:long_name = "runoff discharge" ;
              Qbar:units = "m3.s-1" ;
      double temp_src_time(temp_src_time) ;
              temp_src_time:cycle_length = 0. ;
              temp_src_time:long_units = "days since 1900-01-01" ;
      double salt_src_time(salt_src_time) ;
              salt_src_time:cycle_length = 0. ;
              salt_src_time:long_units = "days since 1900-01-01" ;
```

(continues on next page)

(continued from previous page)

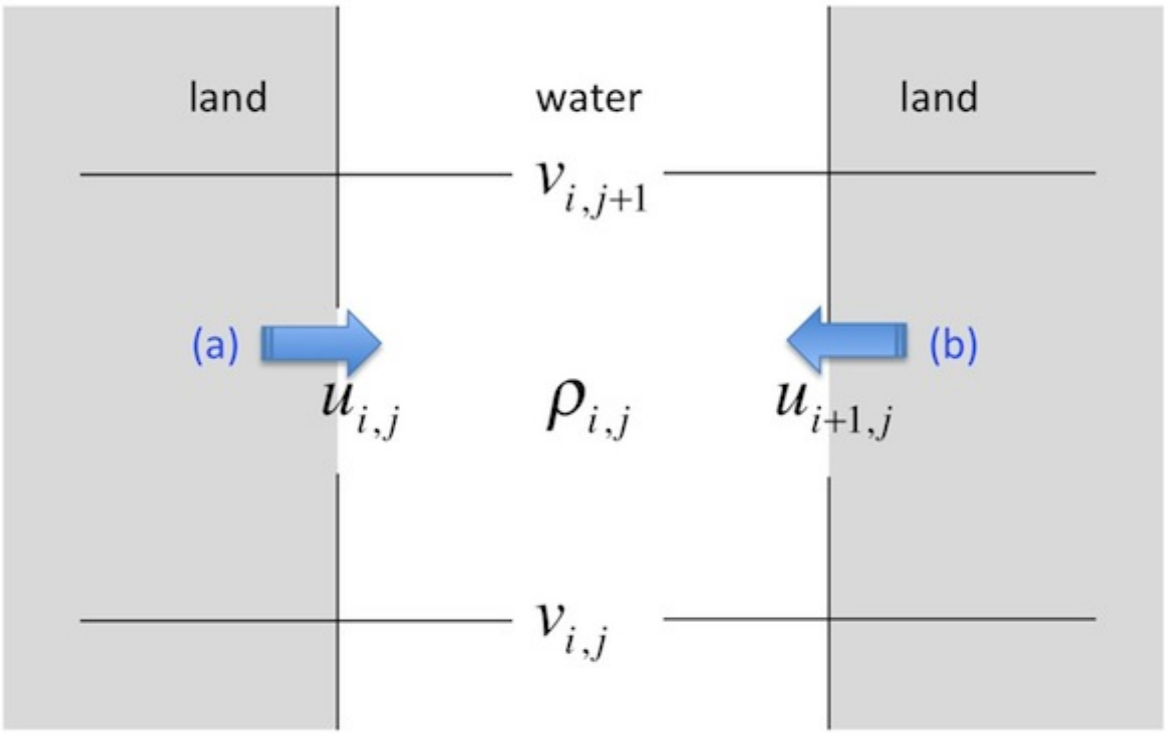
```
      double temp_src(n_qbar, temp_src_time) ;
              temp_src:long_name = "runoff temperature" ;
              temp_src:units = "Degrees Celcius" ;
      double salt_src(n_qbar, temp_src_time) ;
              salt_src:long_name = "runoff salinity" ;
              salt_src:units = "psu" ;
              }
```

When using PSOURCE, Isrc and Jsrc refer to the i,j index of the u-face or v-face the flow crosses - NOT the i,j index of the rho cell it flows into. The i,j values must follow ROMS Fortran numbering convention for the appropriate u-point or v-point on the ROMS staggered grid.

This numbering convention is shown in the figure below (Courtesy of ROMS-RUTGERS team) for flow crossing a u-face into a cell from either the left or the right. This makes it more obvious why the index of the u-face must be specified, because to give the i,j indices of the receiving rho-cell would be ambiguous.

The u-face or v-face should be a land/sea mask boundary (i.e. a coastline). If the cell face is placed wholly in the land you get nothing because there is no wet cell for the flow to enter. If the face is in the middle of open water you have a situation where the flow at that cell face computed by the advection algorithm is 'REPLACED, not augmented, by the source.

It is very easy to misconfigure source/sink locations so caution and careful checking is required.



CHAPTER

THIRTEEN

TIDES

Related CPP options:

TIDES	Activate tidal forcing at open boundaries
SSH_TIDES	process and use tidal sea level data
UV_TIDES	process and use tidal current data
OBC_REDUCED_PHYSICS	compute tidal velcocity from tidal elevation | in case of tidal current is not available
TIDERAMP	Apply ramping on tidal forcing (1 day) at initialization Warning! This should be undefined if restarting the model

Preselected options:

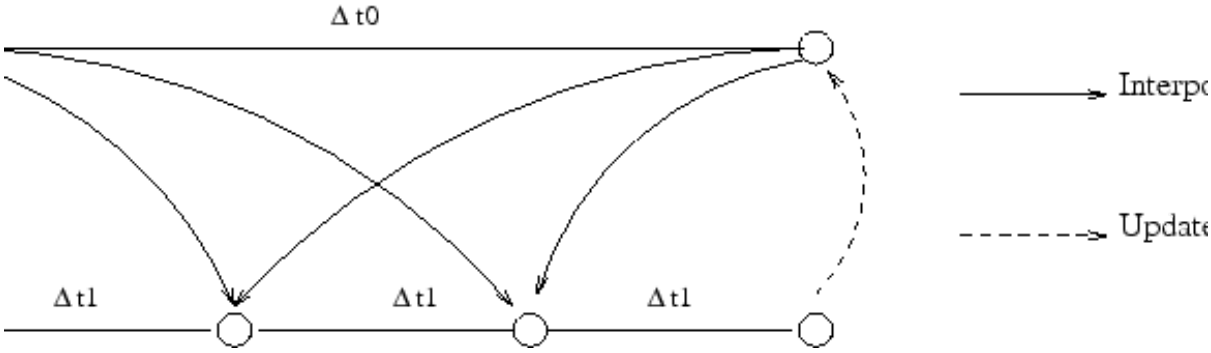
```
# ifdef TIDES
#  define SSH_TIDES
#  define UV_TIDES
#  ifndef UV_TIDES
#   define OBC_REDUCED_PHYSICS
#  endif
#  define TIDERAMP
# endif
```

CHAPTER

FOURTEEN

NESTING CAPABILITIES

To address the challenge of bridging the gap between near-shore and offshore dynamics, a nesting capability has been added to CROCO and tested for the California Upwelling System (Debreu et al., 2012; Penven et al., 2006). The method chosen for embedded griding takes advantage of the AGRIF (Adaptive Grid Refinement in Fortran) package (Debreu and Blayo, 2003, 2008; Blayo and Debreu, 1999; Debreu and Vouland, 2003; Debreu, 2000). AGRIF is a Fortran 95 package for the inclusion of adaptive mesh refinement features within a finite difference numerical model. One of the major advantages of AGRIF in static-grid embedding is the ability to manage an arbitrary number of fixed grids and an arbitrary number of embedding levels.



A recursive integration procedure manages the time evolution for the child grids during the time step of the parent grids (Fig. 2). In order to preserve the CFL criterion, for a typical coefficient of refinement (say, a factor of 3 for a 5 km resolution grid embedded in a 15 km grid), for each parent time step the child must be advanced using a time step divided by the coefficient of refinement as many time as necessary to reach the time of the parent (Fig. 2). For simple 2-level embedding, the procedure is as follows:

1. Advance the parent grid by one parent time step.
2. Interpolate the relevant parent variables in space and time to get the boundary conditions for the child grid.
3. Advance the child grid by as much child time steps as necessary to reach the new parent model time.
4. Update point by point the parent model by averaging the more accurate values of the child model (in case of 2-way nesting).

The recursive approach used in AGRIF allows the specification of any number of nesting levels. Additional CPP options are related to AGRIF, they are in set_global_definitions.h and set_obc_definitions.h files. These are default options intended for nesting developers and should not be edit by standard users.

For a better understanding of ROMS nesting capabilties using AGRIF, check the published articles on CROCO/ROMS nesting implementation and also the AGRIF project homepage:

1. CROCO/ROMS 1 way nesting : Evaluation and application of the ROMS 1-way, embedding procedure to the central california upwelling system, (Penven et al., 2006)
2. CROCO/ROMS 2 way nesting: Two-way nesting in split-explicit ocean models: algorithms, implementation and validation. (Debreu et al., 2012)
3. AGRIF homepage : http://www-ljk.imag.fr/MOISE/AGRIF/

Related CPP options:

AGRIF	Activate nesting capabilities (1-WAY by default)
AGRIF_2WAY	Activate 2-WAY nesting (update parent solution by child solution)

CHAPTER

FIFTEEN

SEDIMENT AND BIOLOGY MODELS

15.1 Bottom Boundary Layer model

Related CPP options:

BBL	Activate bottom boundary layer parametrization
ANA_WWAVE	Set analytical (constant) wave forcing (hs,Tp,Dir).
ANA_BSEDIM	Set analytical bed parameters (if SEDIMENT is undefined)
Z0_BL	Compute bedload roughness for ripple predictor and sediment purposes
Z0_RIP	Determine bedform roughness ripple height and ripple length for sandy bed
Z0_BIO	Determine (biogenic) bedform roughness ripple height and ripple length for silty beds

Preselected options:

```
#ifdef BBL
# ifdef OW_COUPLING
# elif defined WAVE_OFFLINE
# elif defined WKB_WWAVE
# else
#  define ANA_WWAVE
# endif
# ifdef SEDIMENT
#  undef  ANA_BSEDIM
# else
#  define ANA_BSEDIM
# endif
# ifdef SEDIMENT
#  define Z0_BL
# else
#  undef  Z0_BL
# endif
# ifdef Z0_BL
#  define Z0_RIP
# endif
# undef  Z0_BIO
#endif
```

DESCRIPTION

Reynolds stresses, production and dissipation of turbulent kinetic energy, and gradients in velocity and suspended-sediment concentrations vary over short vertical distances, especially near the bed, and can be difficult to resolve with the vertical grid spacing used in regional-scale applications. CROCO provides algorithms to parameterize some of these subgrid-scale processes in the water column and in the bottom boundary layer (BBL). Treatment of the BBL is important for the circulation model solution because it determines the stress exerted on the flow by the bottom, which enters the Reynolds-averaged Navier-Stokes equations as a boundary conditions for momentum in

the x and y directions:

$$K_m \frac{\partial u}{\partial s} = \tau_{bx}$$
$$K_m \frac{\partial v}{\partial s} = \tau_{by}$$

Determination of the BBL is even more important for the sediment-transport formulations because bottom stress determines the transport rate for bedload and the resuspension rate for suspended sediment.

CROCO implements either of two methods for representing BBL processes: (1) simple drag-coefficient expressions or (2) more complex formulations that represent the interactions of wave and currents over a moveable bed. The drag-coefficient methods implement formulae for linear bottom friction, quadratic bottom friction, or a logarithmic profile. The other, more complex wave-current BBL model is described by Blaas et al. (2007) with an example of its use on the Southern California continental shelf. The method uses efficient wave-current BBL computations developed by Soulsby (1995) in combination with sediment and bedform roughness estimates of Grant and Madsen (1982), Nielsen (1986) and Li and Amos (2001).

Linear/quadratic drag

The linear and/or quadratic drag-coefficient methods depend only on velocity components u and v in the bottom grid cell and constant, spatially-uniform coefficients γ_1 and γ_2 specified as input:

$$\tau_{bx} = (\gamma_1 + \gamma_2 \sqrt{u^2 + v^2})\, u$$
$$\tau_{by} = (\gamma_1 + \gamma_2 \sqrt{u^2 + v^2})\, v$$

where γ_1 is the linear drag coefficient and γ_2 is the quadratic drag coefficient. The user can choose between linear or quadratic drag by setting one of these coefficients to zero. The bottom stresses computed from these formulae depend on the elevation of u and v (computed at the vertical mid-elevation of the bottom computational cell). Therefore, in this s-coordinate model, the same drag coefficient will be imposed throughout the domain even though the vertical location of the velocity is different.

Logarithmic drag (with roughness length z_0)

To prevent this problem, the quadratic drag γ_2 can be computed assuming that flow in the BBL has the classic vertical logarithmic profile defined by a shear velocity u_* and bottom roughness length z_0 (m) as:

$$|u| = \frac{u_*}{\kappa} \ln\left(\frac{z}{z_0}\right)$$

where $|u| = \sqrt{u^2 + v^2}$, friction velocity $u_* = \sqrt{\tau_b}$, z is the elevation above the bottom (vertical mid-elevation point of the bottom cell), $\kappa = 0.41$ is von Kármán's constant. z_0 is an empirical parameter. It can be constant (default) or spatially varying. Kinematic stresses are calculated as`

$$\tau_{bx} = \frac{\kappa^2}{\ln^2(z/z_0)} \sqrt{u^2 + v^2} u$$
$$\tau_{by} = \frac{\kappa^2}{\ln^2(z/z_0)} \sqrt{u^2 + v^2} v$$

The advantage of this approach is that the velocity and the vertical elevation of that velocity are used in the equation. Because the vertical elevation of the velocity in the bottom computational cell will vary spatially and temporally, the inclusion of the elevation provides a more consistent formulation.

Combined wave-current drag (BBL)

To provide a more physically relevant value of z_0, especially when considering waves and mobile sediments, a more complex formulation is available (BBL).

The short (order 10-s) oscillatory shear of wave-induced motions in a thin (a few cm) wave-boundary layer produces turbulence and generates large instantaneous shear stresses. The turbulence enhances momentum transfer, effectively increasing the bottom-flow coupling and the frictional drag exerted on the wave-averaged flow. The large instantaneous shear stresses often dominate sediment resuspension and enhance bedload transport. Sediment transport can remold the bed into ripples and other bedforms, which present roughness elements to the flow. Bedload transport can also induce drag on the flow, because momentum is transferred to particles as they are removed

from the bed and accelerated by the flow. Resuspended sediments can cause sediment-induced stratification and, at high concentrations, change the effective viscosity of the fluid.

The BBL parameterization implemented in CROCO requires inputs of velocities u and v at reference elevation z, representative wave-orbital velocity amplitude u_b, wave period T, and wave propagation direction θ (degrees, clockwise from north). The wave parameters may be the output of a wave model such as WKB or WW3 or simpler calculations based on specified surface wave parameters. Additionally the BBL models require bottom sediment characteristics (median grain diameter D_{50}, mean sediment density ρ_s, and representative settling velocity w_s); these are constant (ANA_BSEDIM) or based on the composition of the uppermost active layer of the bed sediment during the previous time step if the sediment model is used.

The wave-averaged, combined wave–current bottom stress is expressed as function of τ_w and τ_c (i.e., the stress due to waves in the absence of currents and due to currents in the absence of waves, respectively) according to Soulsby (1995):

$$\bar{\tau}_{wc} = \tau_c \left(1 + 1.2 \left(\frac{\tau_w}{\tau_w + \tau_c} \right)^{3.2} \right)$$

The maximum wave–current shear stress within a wave cycle is obtained by adding $\bar{\tau}_{wc}$ and τ_w (with ϕ the angle between current and waves):

$$\tau_{wc} = \left((\bar{\tau}_{wc} + \tau_w \cos\phi)^2 + (\tau_w \sin\phi)^2 \right)^{1/2}$$

The stresses τ_c and τ_w are determined using:

$$\tau_c = \frac{\kappa^2}{\ln^2(z/z_0)} |u|^2$$
$$\tau_w = 0.5 \rho f_w u_b^2$$

u_b, the bottom orbital velocity, is determined from the significant wave height H_s and peak frequency ω_p using the Airy wave theory:

$$u_b = \omega_p \frac{H_s}{2 \sinh kh}$$

with h the local depth and k the local wave number from the dispersion relation. The wave-friction factor f_w is, according to Soulsby (1995):

$$f_w = 1.39 (u_b / \omega_p z_0)^{-0.52}$$

The wave–current interaction in the BBL is taken into account only if $u_b > 1$ cm/s; otherwise, current-only conditions apply.

Shear stress for sediment resuspension and roughness length due to bed form

To determine the shear stress relevant for sediment resuspension and the roughness length due to bed forms, we follow the concept of Li and Amos (2001) briefly summarized here. First, the maximum wave–current skin friction τ_s is computed from the equations above, using the Nikuradse roughness $z_0 = D_{50}/12$.

A bed-load layer develops as soon as the maximum wave–current skin friction τ_s exceeds the critical stress τ_{cr}. This layer affects the stress effective for ripple formation and sediment resuspension. Subsequently, for sandy locations, ripple height and length are computed, leading to a space- and time-dependent ripple roughness length $z_0 = z_{rip}$, which is used to compute the drag on the flow (instead of a constant value when BBL is not activated). This drag provides boundary conditions to the momentum and turbulence equations (KPP or GLS).

15.2 Sediment models

There are two sediment models in CROCO: the USGS model derived from the UCLA/USGS ROMS community, and MUSTANG derived from the Ifremer SIAM/MARS community.

15.2.1 USGS Sediment Model

This USGS sediment model is derived from the UCLA/USGS ROMS community. See Blaas et al. (2007), Warner et al. (2008) and Shafiei et al. (2021) for details.

Regarding the time and space resolution considered, the explicit solution generally refers to quantities averaged over wave periods, although the implementation of a nonhydrostatic solver in CROCO opens the way to a wave-resolved approach. One of the crucial ingredients in the sediment transport model is a reliable representation of wave-averaged (or wave-resolved) hydrodynamics and turbulence.

In the wave-averaged approach, the wave boundary layer is not resolved explicitly, but the lower part of the velocity and sediment concentration profile in the current boundary layer is important for the calculation of the sediment transport rates. Similarly, an accurate assessment of the bottom boundary shear stress (including wave effects) is required since it determines the initiation of grain motion and settling and resuspension of suspended load (see BBL). Thus, the sediment concentration and current velocity profiles in the unresolved part of the near-bottom layer have to be parameterized. Characterization of the sediments (mainly density and grain size, making general assumptions about shape and cohesiveness) is done either as a time-dependent prescribed function at the point sources or at the sea bed as an initial (soon space-dependent) condition. Sediment concentration may be considered as passive with respect to the flow density or as active if concentration values require such (the latter is not implemented yet).

Sediment bed

The sediment bed is represented by three-dimensional arrays with a fixed number of layers beneath each horizontal model cell. Each cell of each layer in the bed is initialized with a thickness, sediment-class distribution, porosity, and age. The mass of each sediment class in each cell can be determined from these values and the grain density. The bed framework also includes two-dimensional arrays that describe the evolving properties of the seabed, including bulk properties of the surface layer (active layer thickness, mean grain diameter, mean density, mean settling velocity, mean critical stress for erosion) and descriptions of the subgrid scale morphology (ripple height and wavelength). These properties are used to estimate bed roughness in the BBL formulations and feed into the bottom stress calculations. The bottom stresses are then used by the sediment routines to determine resuspension and transport, providing a feedback from the sediment dynamics to the hydrodynamics.

The bed layers are modified at each time step to account for erosion and deposition and track stratigraphy. At the beginning of each time step, an active layer thickness z_a is calculated (Harris and Wiberg, 1997). z_a is the minimum thickness of the top bed layer. If the top layer is thicker than z_a, no action is required. If the top layer is less than z_a, then the top layer thickness is increased by entraining sediment mass from deeper layers until the top layer thickness equals z_a. If sediment from deeper than the second layer is mixed into the top layer, the bottom layer is split to enforce a constant number of layers and conservation of sediment mass. Each sediment class can be transported by suspended-load and/or bedload (below). Suspended-load mass is exchanged vertically between the water column and the top bed layer. Mass of each sediment class available for transport is limited to the mass available in the active layer. Bedload mass is exchanged horizontally within the top layer of the bed. Mass of each sediment class available for transport is limited to the mass available in the top layer. Suspended-sediment that is deposited, or bedload that is transported into a computational cell, is added to the top bed layer. If continuous deposition results in a top layer thicker than a user-defined threshold, a new layer is provided to begin accumulation of depositing mass. The bottom two layers are then combined to conserve the number of layers. After erosion and deposition have been calculated, the active-layer thickness is recalculated and bed layers readjusted to accommodate it. This step mixes away any very thin layer (less than the active layer thickness) of newly deposited material. Finally the surficial sediment characteristics, such as D50, ripple geometry, etc., are updated and made available to the bottom stress calculations.

Suspended-sediment transport

The concentration of sediment suspended in the water column is transported, like other conservative tracers (e.g., temperature and salinity) by solving the advection–diffusion equation with a source/sink term for vertical settling and erosion:

$$\underbrace{\frac{\partial C}{\partial t}}_{RATE} = - \underbrace{\vec{\nabla}.\vec{\mathbf{v}}C}_{ADVECTION} + \underbrace{\mathcal{D}_C}_{MIXING} - \underbrace{\frac{\partial w_s C}{\partial z}}_{SETTLING} + \underbrace{\frac{E}{\delta z_b}\Big|_{z=z_b}}_{EROSION}$$

C is the Reynolds-averaged, wave-averaged (unless used in wave-resolving mode) sediment concentration of a particular size class; $\vec{\mathbf{v}}$ is the flow velocity (it is the Lagrangian velocity $\vec{\mathbf{v}_L}$ in wave-averaged equations, comprising the Stokes drift $\vec{\mathbf{v}_S}$).

For each size class, the source or sink term represents the net of upward flux of eroded material E and downward settling, i.e., the deposition flux. w_s is the settling velocity, dependent on sediment grain size, but independent of flow conditions and concentrations. It is an input parameter of the model (WSED in sediment.in; see below). Settling is computed via a semi-Lagrangian advective flux algorithm, which is unconditionally stable (Durran, 2010). It uses a piece-wise parabolic vertical reconstruction of the suspended sediment for high-order interpolation, with WENO constraints to avoid oscillations. E is the erosion flux at the sea floor and is only applied to the first grid level of height z_b and cell size δz_b. The erosion flux for each class is given by:

$$E = E_0(1-p)\,\phi\left(\frac{\tau_s}{\tau_c} - 1\right) \text{ for } \tau_s > \tau_c$$

E_0 is an empirical erosion rate (ERATE parameter in sediment.in; see below); p is the sediment porosity; ϕ is the volumetric fraction of sediment of the class considered; τ_c is the critical shear stress; and τ_s is the shear stress magnitude on the grains (skin stress due to wave-induced bed orbital velocities and mean bottom currents; see BBL). The critical shear stress is the threshold for the initiation of sediment motion.

Zero-flux boundary conditions are imposed at the surface and bottom in the vertical diffusion equation. Lateral open boundaries are treated as other tracers according to Marchesiello et al. (2001). A quasi-monotonic 5th-order advection scheme (WENO5-Z, Borges et al., 2008) can be used for horizontal and vertical advection of all tracers, including sediments.

Bedload transport

The bedload flux q_b, which is considered unresolved by the model can be calculated using different bedload models implemented in CROCO. The formulation by Meyer-Peter Muller (Meyer-Peter and Muller, 1948) is suited to rivers or continental shelf problems, where nonlinear wave effects are small. For nearshore applications, where wave nonlinearity is important, the bedload transport formulation proposed by van der A et al. (2013) is implemented as in Shafiei et al. (2021) following Kalra et al. (2019) with some modifications.

Each formulation depends on the characteristics of individual sediment classes, including median size d_{50}, grain density ρ_s, specific density in water $s = \rho/\rho_s$, and critical shear stress τ_c. Non-dimensional transport rates Φ are calculated for each sediment class and converted to dimensional bedload transport rates q_b using:

$$q_b = \Phi\sqrt{(s-1)g d_{50}^3 \rho_s}$$

These are horizontal vector quantities with directions that correspond to the combined bed-stress vectors. Details on the computation of Φ differs in the Meyer-Peter Müeller or van der A formulations.

Slope effect: bedload fluxes are corrected to account for the avalanche process, i.e., the gravitational flow of sand occuring when the bottom slope exceeds the critical slope angle:

$$q_{b,slope} = q_b\left(\frac{0.65}{(0.65 - \tan\beta)\cos\beta}\right),$$

This correction considers the effect of the bed slope $\beta = \tan^{-1}(dz_b/dx)$. The value 0.65 is derived from the consideration of an angle of repose of 33°.

Bedload numerics: bedload fluxes are computed at grid-cell centers and are limited by the availability of each sediment class in the top layer. Fluxes are then interpolated on cell faces using an upwind approach, either 1rst-order (e.g., Lesser et al., 2004) or 5th order, or even a WENO5 interpolation to avoid oscillations. Flux differences are then used to determine changes of sediment mass in the bed at each grid cell.

Meyer-Peter Müeller (1948) : Transport by currents

$$\Phi = max\left[8(\theta_s - \theta_c)^{1.5}, 0\right]$$

where Φ is the magnitude of the non-dimensional transport rate for each sediment class, θ_s is the non-dimensional Shields parameter for skin stress:

$$\theta_s = \frac{\tau_s}{(s-1)gd_{50}}$$

θ_c is the critical Shields parameter, and τ_s the magnitude of total skin-friction component of bottom stress computed from:

$$\tau_s = \sqrt{\tau_{sx}^2 + \tau_{sy}^2}$$

where τ_{sx} and τ_{sy} are the skin-friction components of bed stress, from currents alone or the maximum wave-current combined stress, in the x and y directions. These are computed at cell faces (u and v locations) and then interpolated to cell centers (ρ points). The bedload transport vectors are partitioned into x and y components based on the magnitude of the bed shear stress as:

$$q_{bx} = q_b \frac{\tau_{sx}}{\tau_s}$$
$$q_{by} = q_b \frac{\tau_{sy}}{\tau_s}$$

van der A (2013): Transport by nonlinear waves

The SANTOSS bedload model is based on the half-wave cycle concept proposed by Dibajnia and Watanabe (1992) that captures asymmetric transport by non-linear waves and the effect of phase lag between mobilization and transport. CROCO contains an adapted version by Shafiei et al. (2021), based on the implementation of Kalra et al. (2019). In our formulation, the effect of wave-averaged currents is removed by default (assuming that the transport by currents is effectively performed by the suspended load model) and it thus only retains the nonlinear effects of waves. In the brief presentation below, we retain the current terms for completeness, but focus on wave effects.

The method to obtain bedload transport under asymmetric waves can be divided into three major steps (van der A, 2013; Kalra et al, 2019). In the first step, the asymmetric waveform based on the Ursell number is evaluated using wave statistics. The Shields parameter for each half cycle of the wave form is computed in the second step. Finally, a phase lag is estimated from the velocity and sediment concentrations that determine the amount of bedload transported in the half cycle following mobilization. The non-dimensional bedload transport rate Φ is thus given by:

$$\Phi = \frac{1}{T}\left[\frac{\theta_c}{|\theta_c|^{1/2}} T_c \left(\Omega_{cc} + \frac{T_c}{2T_{cu}}\Omega_{tc}\right) + \frac{\theta_t}{|\theta_t|^{1/2}} T_t \left(\Omega_{tt} + \frac{T_t}{2T_{tu}}\Omega_{ct}\right)\right],$$

where T, T_c, T_t, T_{cu} and T_{tu} are the wave period, duration of wave crest half cycle, duration of wave trough half cycle, duration of accelerating flow within the crest half cycle and duration of accelerating flow within the trough half cycle respectively, θ_c and θ_t represent the Shields numbers associated with the wave crest and trough half cycles. The sand load transported during the crest period is the combination of Ω_{cc} (mobilized during the crest period) and Ω_{tc} (mobilized during the trough period). Similarly, Ω_{tt} and Ω_{ct} are the sand load transported during the trough period (mobilized during the trough and crest periods respectively).

The sand load transported during each half-cycle is conventionally modeled according to a power law of Shields number:

$$\Omega_i = \max\left(11\left(|\theta_i| - \theta_{cr}\right)^{1.2}, 0\right),$$

where θ_{cr} is the critical Shields number and, hereafter, the subscript "i" is either "c" for crest or "t" for trough half cycles. To determine Ω_{ct} and Ω_{tc}, i.e., the portion of the bedload remaining in suspension to be transported in the next half cycle, a phase lag parameter is evaluated.

Let's assume a two-dimensional (x,z) cross-shore problem for simplicity. The Shields number for the peak or trough ($\theta_i = \theta_t$ or θ_c) is calculated according to:

$$\theta_i = \frac{\frac{1}{2} f_{w\delta i} |u_{i,r}| u_{i,r}}{(s-1) g d_{50}}.$$

$u_{i,r}$ is the representative cross-shore combined wave-current velocity at trough or crest half cycles calculated as:

$$u_{i,r} = \frac{\hat{u}_i}{\sqrt{2}} + |u_\delta|,$$

where $\hat{u}_i$ is the peak crest or trough orbital velocities, u_δ is the steady current velocity at the top of the wave boundary layer. $f_{w\delta i}$ is the linear wave-current friction factor at crest or trough calculated by Ribberink (1998}:

$$f_{w\delta i} = \frac{\hat{u}}{u_\delta + \hat{u}} f_{wi} + \frac{u_\delta}{u_\delta + \hat{u}} f_\delta$$

where $\hat{u}$ is the representative orbital velocity amplitude for the whole flow cycle (given by $\hat{u} = \sqrt{2} u_{orb}$). f_δ is the current-related friction factor dependent on a current-related roughness $k_{s\delta}$ and f_{wi} is the wave friction factor, calculated separately for the crest and trough half-cycles and depends on a wave-related roughness k_{sw}. If the representative orbital excursion amplitude $\hat{a} = \hat{u}T/2\pi$ is large enough (i.e., greater than 1.587 k_{sw}):

$$f_{wi} = 0.00251\, e^{5.21\left[\left(\frac{2T_{iu}}{T_i}\right)^{2.6} \frac{\hat{a}}{k_{sw}}\right]^{-0.19}},$$

otherwise, $f_{wi} = 0.3$.

If $u_\delta = 0$ ($u_{i,r} = \hat{u}_i/\sqrt{2}$ and $f_{w\delta i} = f_{wi}$), the effect of currents is completely removed from the bedload transport calculation. This choice represents our default to avoid double counting the transport by wave-averaged currents.

Finally, following Kalra et al. (2019), after calculating Φ, we apply to q_b a bedload factor f_{bld} ($bedload_coeff$ in CROCO). This factor allows us to adjust the relative contribution to sediment transport of wave-induced bedload compared to the suspended load transported by mean currents. In this way, we have a better control of the antagonistic mechanisms that govern onshore and offshore transports respectively.

Morphology

The bed evolution (variation in time of z_b, the height of the bed), is calculated from the divergence of sediment fluxes (Exner equation), which results from the difference between erosion and sedimentation of suspended sediments. In wave-averaged equations, where residual wave effects need to be parametrized as bedload fluxes q_b, the bed evolution also arises from the divergence of these fluxes.

$$\frac{\partial z_b}{\partial t} = -\frac{f_{mor}}{1-p}\left(\frac{\partial q_b}{\partial x} - w_s \frac{\partial C}{\partial z} + E\right).$$

This equation accounts for a morphological acceleration factor f_{mor} ($morph_fac$ in CROCO). A value of 1 has no effect, and values greater than 1 accelerate the bed response. The concept of morphological acceleration is based on the fact that morphodynamic changes are slower than hydrodynamic ones (van Rijn, 1993). In this case, the bed evolution can be accelerated without affecting the hydro-morphological solution. The increased rate of morphological change can be useful for simulating evolution over long time periods. Strategies for morphological updating are described by Roelvink (2006) and implemented in CROCO following Warner et al. (2008). In our implementation, bedload fluxes, erosion, and deposition rates are multiplied by f_{mor}, while the magnitude of sediment concentrations in the water column is not modified – just the exchange rate to and from the bed. For

both bedload and suspended load, sediment is limited in availability, based on the true amount of sediment mass (not multiplied by the scale factor).

For dynamical consistency, the vertical velocity is modified (in omega.F) by the rate of change of vertical grid levels dz/dt, adjusting to the moving sea floor and free surface (grid "breathing" component; Shchepetkin and McWilliams, 2005). This method is mass conserving and retains tracer constancy preservation.

Sediment Density

This is not implemented yet. Effects of suspended sediment on the density field can be included with terms for the weight of each sediment class in the equation of state for seawater density as:

$$\rho = \rho_{water} + \sum_{m=1}^{Nsed} \frac{C_m}{\rho_{s,m}} \left(\rho_{s,m} - \rho_{water}\right)$$

This enables the model to simulate processes where sediment density influences hydrodynamics, such as density stratification and gravitationally driven flows.

Related CPP options:

SUSPLOAD	Activate suspended load transport
BEDLOAD	Activate bedload transport
MORPHODYN	Activate morphodynamics
BEDLOAD_VANDERA	van der A formulation for bedload (van der A et al., 2013)
BEDLOAD_MPM	Meyer-Peter-Muller formulation for bedload (Meyer-Peter and Muller, 1948)
SLOPE_LESSER	Lesser formulation for avalanching (Lesser et al, 2004)
SLOPE_NEMETH	Nemeth formulation for avalanching (Nemeth et al, 2006)
BEDLOAD_UP1	Bedload flux interpolation: upwind 1rst order
BEDLOAD_UP5	Bedload flux interpolation: upwind 5th order
BEDLOAD_WENO5	Bedload flux interpolation: WENO 5th order
ANA_SEDIMENT	Set analytical sediment size, initial ripple and bed parameters
ANA_BPFLUX	Set kinematic bottom flux of sediment tracer (if different from 0)
SPONGE_SED	Gradually reduce erosion/deposition near open boundaries

Preselected options:

```
#ifdef SEDIMENT
# undef  MUSTANG
# define ANA_SEDIMENT
# define SPONGE_SED
# define Z0_BL
# define Z0_RIP
# ifdef BEDLOAD
#  ifdef BEDLOAD_VANDERA       /* default BEDLOAD scheme  */
#  elif defined BEDLOAD_MPM
#  elif defined BEDLOAD_WULIN
#  elif defined BEDLOAD_MARIEU
#  else
#   if (defined WAVE_OFFLINE || defined WKB_WWAVE ||\
        defined ANA_WWAVE    || defined OW_COUPLING)
#    define BEDLOAD_VANDERA
#   else
#    define BEDLOAD_MPM
#   endif
#  endif
#  ifdef BEDLOAD_UP1           /* default INTERPOLATION  */
#  elif defined BEDLOAD_UP5
```

(continues on next page)

(continued from previous page)

```
#  elif defined BEDLOAD_WENO5
#  else
#   define BEDLOAD_UP1
#  endif
#  ifdef SLOPE_LESSER          /* default SLOPE scheme */
#  elif defined SLOPE_NEMETH
#  elif defined SLOPE_KIRWAN
#  else
#   define SLOPE_LESSER
#  endif
# endif /* BEDLOAD */
 #endif /* SEDIMENT */
```

Parameters in sediment.in

```
1  Stitle (a80)
CROCO - Sediment - Test

2  Sd(1-NST), CSED, SRHO,  WSED,  ERATE,  TAU_CE, TAU_CD, BED_FRAC(1:NLAY)
     0.125    9.9   2650.  9.4    25.0e-5  0.05    0.14    0.4  0.4
     0.050    0.0   2650.  1.6     4.0e-5  0.01    0.14    0.6  0.6

3 BTHK(1:NLAY)
      1.    10.

4 BPOR(1:NLAY)
      0.41  0.42

5 Hrip
      0.03

6 Lrip
      0.14

7 bedload_coeff
      0.

8 morph_fac
     10.

9  transC
     0.03

10  transN
     0.2

11  tcr_min
     0.03

12  tcr_max
     5.5

13  tcr_slp
     0.3

14  tcr_off
     1.

15  tcr_tim
     28800.
```

(continues on next page)

(continued from previous page)

```
16  L_ADS  L_ASH  L_COLLFRAG  L_TESTCASE
    F       T         F           F

17  F_DP0      F_NF    F_DMAX     F_NB_FRAG    F_ALPHA   F_BETA  F_ATER   F_ERO_
↪FAC F_ERO_NBFRAG  F_COLLFRAGPARAM  F_CLIM    F_ERO_IV
   0.000004    2.     0.0015        2.          0.35     0.15    0.          0.
↪   2.             0.01          0.001        1

18  MUD_FRAC_EQ [1:NMUD]
   0.10  0.20  0.40  0.20  0.10  0.0  0.0  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

19  MUD_T_DFLOC
   200.

99 END of sediment input data
```

GLOSSARY

CARD 1: String with a maximum of eighty characters.

> **Stitle** : Sediment case title.

CARD 2: Sediment grain parameters & initial values (NST lines)

> **Sd** : Diameter of grain size class [mm].

> **CSED** : Initial concentration (spatially uniform) [kg/m3].

> **SRHO** : Density of sediment material of size class [kg/m3].

Quartz: SRHO=2650 kg/m^3

> **WSED** : Settling velocity of size class [mm/s].

Typically (Soulsby, 1997):

WSED = 10^3 $(visc$ $(\sqrt{10.36^2 + 1.049D^3}$ $- 10.36)$ / D_{50} [mm/s]

with $D = D_{50}$ $(g$ $(\mathrm{SRHO}/\rho_0 - 1)$ / $(visc^2)$ $)^{0.33333}$

$D_{50} = 10^{-3} Sd$ [m]

$visc = 1.3\ 10^{-3}/\rho_0$ [m2/s]

> **ERATE** : Erosion rate of size class [kg/m2/s].

Typically:

ERATE $= 10^{-3}$ γ_0 WSED SRHO $[kg/m^2/s]$

with $\gamma_0 = 10^{-3} - 10^{-5}$ (Smith & McLean, 1977)

> **TAU_CE** : Critical shear stress for sediment motion [N/m2]

(initiation of bedload for coarses, suspension for fines). Typically : $\mathrm{TAU_CE} = 6.4\ 10^{-7}$ ρ_0 WSED^2 [N/m2]

> **TAU_CD** : Critical shear stress for deposition of cohesive sediments [N/m2]

> **BED_FRAC** : Volume fraction of each size class in each bed layer (NLAY columns)

[0<BED_FRAC<1]

CARD 3: Sediment bed thickness, 1st field is top layer ('delt_a')

> **BTHK** : Initial thicknesses of bed layers [m]

Bthk(1) active layer thickness, fixed in simulation unless SUM(Bthk(:))<Bthk(1)

CARD 4: Sediment bed porosity

> **BPOR** : Initial porosity of bed layers [m]

used in ana_sediment ifdef ANA_SEDIMENT (not in init.nc)

CARD 5: Bottom ripple height

> **Hrip** : Initial ripple height [m]

used in ana_sediment ifdef ANA_SEDIMENT (not in init.nc)

CARD 6: Bottom ripple length

> **Lrip** : Initial ripple length [m]

used in ana_sediment ifdef ANA_SEDIMENT (not in init.nc)

CARD 7: Bedload coefficient

> **bedload_coeff** : factor limiting the magnitude of bedload flux

0<bedload_coef<1

CARD 8: Morphological acceleration factor

> **morph_fac** : factor accelerating bed evolution

morph_fac>=1

Card 9 :

> **transC** : Cohesive transition- Under that value of total mud fraction

entire bed behaves as a non-cohesive bed

Card 10 :

> **transN** : Noncohesive transition- Over that value of total mud fraction

entire bed behaves as a cohesive bed

Card 11 :

> **tcr_min** : Minimum shear for erosion

Card 12 :

> **tcr_max** : Maximum shear for erosion

Card 13 :

> **tcr_slp** : Tau_crit profile slope

Card 14 :

> **tcr_off** : Tau_crit profile offset

Card 15 :

> **tcr_tim** : Tau_crit consolidation rate

Card 16 :

booleans for flocculation

> **L_ADS** : Boolean set to .true. if differential settling aggregation

> **L_ASH** : Boolean set to .true. if shear aggregation

> **L_COLLFRAG** : Boolean set to .true. if collision-induced fragmentation enable

> **L_TESTCASE** : If .TRUE. sets G(t) to values from Verney et al., 2011 lab experiment

---

Card 17 :

flocculation Sediment Parameters.

> **F_DP0** : Primary particle size (m), typically 4e-6 m

> **F_NF** : Floc fractal dimension, typically ranging from 1.6 to 2.6

> **F_DMAX** : Maximum diameter (m)

References

Blaas, M., Dong, C., Marchesiello, P., McWilliams, J.C., Stolzenbach, K.D., 2007. Sediment-transport modeling on southern californian shelves: A roms case study. Continental shelf research 27, 832–853.

Warner, J.C., Sherwood, C.R., Signell, R.P., Harris, C.K., Arango, H.G., 2008. Development of a three- dimensional, regional, coupled wave, current, and sediment-transport model. Computers & geosciences 34, 1284–1306.

Kalra, T., Sherwood, C., Warner, J., Rafati, Y., Hsu, T.J., 2019. Investigating bedload transport under asymmetrical waves using a coupled ocean-wave model. pp. 591–604.

Shafiei H., J. Chauchat, C. Bonamy, and P. Marchesiello, 2021: Numerical simulation of on-shore/off-shore sandbar migration using wave-cycle concept – application to a 3D wave-averaged oceanic model (CROCO), submitted to Ocean Modelling.

15.2.2 MUSTANG Sediment model

MUSTANG presentation

MUSTANG (**MU**d and **S**and **T**ran **ANS**sport modelli **NG**) is a sediment module. Comparing to a pure hydrodynamical simulation, it adds variables relatives to sediment behavior such as sediment concentration in water and sediment bed evolution. This module has been developped by **Le Hir et al**, coupled with the hydrodynamic model **SiAM**. Then, it has continued to evolve, coupled with the parallelized **MARS3D** hydrodynamic model. It is now renamed **MUSTANG** (MUd and Sand TranSport modelliNG) and is available in the **CROCO** community model since release 1.2.

The module provides for the modeling of two types of sediment transport :

- **Bedload** : sediment particles are set in motion from a certain critical tension. They move by staying close to the bottom
- **Suspension** : sediment particles are suspended in the water column. They are transported in the water column as a passive tracer with a settling velocity through the advection-diffusion equation and eventually settle to the bottom.

MUSTANG module deals with various sediments classified into 3 types: GRAVEL, SAND and MUD. Gravels are transported only in bedload (no suspension). Sands can be transported by both types. Muds are transported only in suspension (no bedload).

From a sedimentary bottom consisting of a set of sedimentary facies, each of which is defined by a proportion of each of the classes, hydrosedimentary modeling consists of changing the proportions of each class of the sedimentary bottom as well as the thickness of the layers which constitute it, and therefore the bathymetry. It is then possible to inject the new bathymetry thus obtained into the hydrodynamic module (morphodynamic coupling).

MUSTANG computes these evolutions by taking into account several processes such as settling, flocculation, erosion fluxes, deposit fluxes, consolidation (porosity) in sediment bed, bedload transport for non-cohesive sediment, morphodynamic...

There are 2 main options for this module :

- one equivalent to the previous module "mixsed" (Le Hir et al, 2011) - **default**
- one developped by Mengual & Le Hir (2018), which includes bedload processes (Rivier et al, 2017) - **activated by cppkey #key_MUSTANG_V2**

MUSTANG sediment compartment is 1DV in the sediment part, all exchanges between horizontal meshes, such as bedload, lateral erosion or sliding effect, are done at the interface between sediment and water.

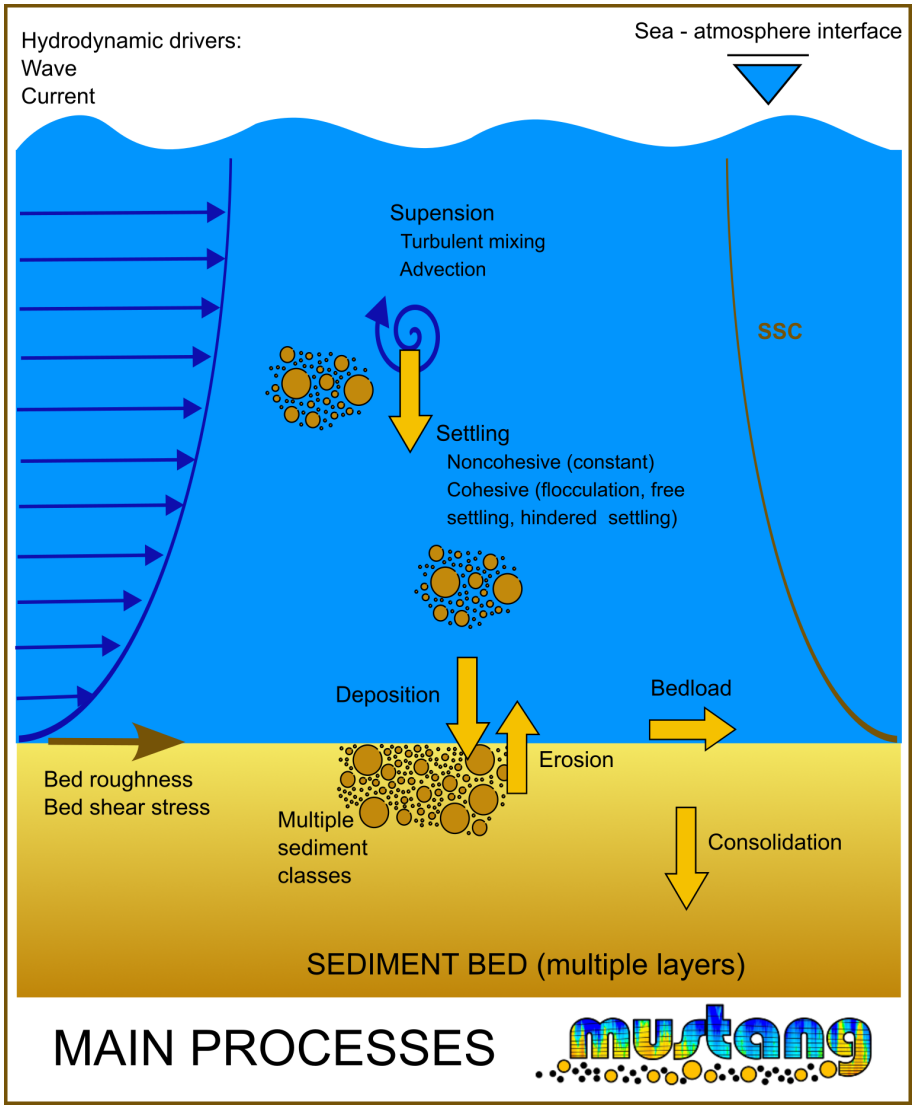
MUSTANG consider 3 types of sediment but all features are not available for all type of sediment :

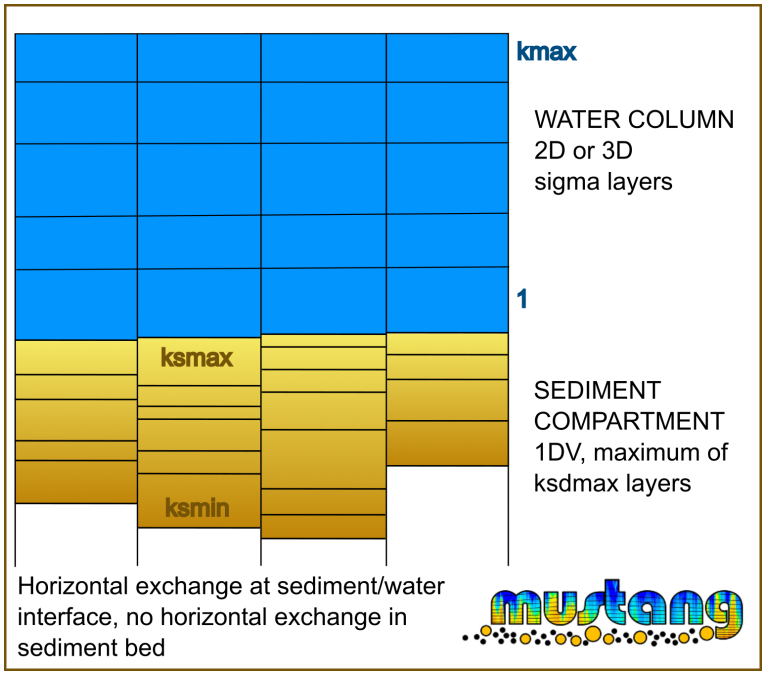
Type	Bedload transport (if #key_MUSTANG_V2 & #key_MUSTANG_bedload)	Suspended load	Flocculation (if #key_MUSTANG_flocmod)
MUD	NO	YES	YES
SAND	YES	YES	NO
GRAVEL	YES	NO	NO

Other substances can be defined via the susbstance module and can interact with sediment depending on their characteristics.

This documentation presents :

- in MUSTANG *user guide*, input and output files format are detailed and available cppkeys are listed with a description of the effect of each key and a list of compatibility/incompatibility between keys
- in MUSTANG *technical documentation*, a description of the insertion of MUSTANG calls in the CROCO temporal loop and a description of the modelisation of the main processes
- in MUSTANG *references* the bibliography


Hydrodynamic drivers:
Wave
Current
Sea - atmosphere interface
Supension
Turbulent mixing
Advection
SSC
Settling
Noncohesive (constant)
Cohesive (flocculation, free settling, hindered settling)
Deposition
Bedload
Erosion
Bed roughness
Bed shear stress
Multiple sediment classes
Consolidation
SEDIMENT BED (multiple layers)
MAIN PROCESSES
mustang


kmax
WATER COLUMN
2D or 3D
sigma layers
1
ksmax
ksmin
SEDIMENT
COMPARTMENT
1DV, maximum of
ksdmax layers
Horizontal exchange at sediment/water interface, no horizontal exchange in sediment bed
mustang

MUSTANG user guide

MUSTANG module can not be used alone, it needed an hydrodynamic model and the substance module.

In this documentation, the hydrodynamic model is **CROCO**. Hydrodynamic parametrization will not be detailed here, please refer to hydrodynamic module documentation.

To use MUSTANG in CROCO you will need to :

- define appropriate cppkeys in **cppdefs.h**. To activate MUSTANG within CROCO environment, the **MUSTANG** and **SUBSTANCE** cppkeys must be defined. To activate certains processes other cppkeys must be defined (see *available cppkeys*)
- define MUSTANG and SUBSTANCE files in CROCO input file **croco.in**, these file contains MUSTANG parameters and SUBSTANCE characteristics (see *MUSTANG namelist* and *SUBSTANCE namelist*)
- define appropriate dimensions in **param.h** file
- depending on the options you choose via your cppkeys and input file (MUSTANG namelist and SUBSTANCE namelist), you will have to define other input file such as *initialization file*, *wave file*, *source with solid discharge file*

Then you can compile and run CROCO as any other CROCO run.

Note: MUSTANG is controlled through both CPP keys and input files. For some processes it is needed to activate the options through a CPP key, and also through a flag (true or false) in the input files

---

Input file : croco.in

The name and location of the MUSTANG and SUBSTANCE input files are defined in CROCO input file **croco.in** as follow:

```
sediments_mustang: input file
                MUSTANG_NAMELIST/parasubstance.txt
                MUSTANG_NAMELIST/paraMUSTANG.txt
```

In this example, the files are located in MUSTANG_NAMELIST directory but they could be placed in any directory. Therefore, example and test cases namelists could be found in MUSTANG_NAMELIST directory in MUSTANG source directory.

In particular, in MUSTANG_NAMELIST directory (relative path from the current directory where croco executable is), a file **paraMUSTANG_default.txt** contains all default values used, if the user does not specify a parameter in the namelist file, the default value in MUSTANG_NAMELIST/paraMUSTANG_default.txt is used.

There is no default file for susbstance but a full example is given in MUSTANG_NAMELIST directory in MUSTANG source directory.

Input file : param.h

Before the code compilation, the following dimensions must be defined in the **param.h** file to use SUBSTANCE and MUSTANG :

- ksdmin and ksdmax: integers corresponding to the layers of sediment (sediment variables are allocated with ksdim:ksdmax dimension)
- ntrc_subs: number of substance corresponding to a tracer (advected)
- ntfix: number of fixed substance (not advected)
- ntrc_substot: total number of substance (= ntrc_subs + ntfix)

If these dimensions are modified, the code must be compiled again.

Input file : Substance namelist

Substance module allow to define 8 different types of substance with different behavior. Sediments are defined as MUD, SAND or GRAVEL.

Substance input file contains at least 5 namelists and at most 9 namelists depending on the cppkeys MUSTANG and key_benthic :

- *nmlnbvar* : number of each type of substance to be defined (other than T (temperature) & S (salinity))
- *nmlpartnc* : characterization of Non Constitutive Particulate subtances
- *nmlpartsorb* : characterization of particulate susbtances sorbed on an other particule
- *nmlvardiss* : characterization of dissolved susbtances
- *nmlvarfix* :characterization of fixed susbtances (not advected)
- *nmlgravels* (if MUSTANG only) : characterization of GRAVEL substances
- *nmlsands* (if MUSTANG only) : characterization of SAND substances
- *nnmlmuds* (if MUSTANG only) : characterization of MUD substances
- *nmlvarbent* (if key_benthic only) : characterization of benthic substances

Each namelist is described bellow with the description of each parameter.

&nmlnbvar namelist:

- **nv_dis** : number of dissolved susbtances
- **nv_ncp**: number of Non Constitutive Particulate subtances, like for example trace metals, bacteria, nitrogen, organic matter.
- **nv_fix** : number of fixed susbtances (not advected)
- **nv_bent** : number of benthic susbtances
- **nv_grav** : number of Constitutive susbtances type GRAVELS (only if cppkey MUSTANG)
- **nv_sand** : number of Constitutive susbtances type SAND (only if cppkey MUSTANG)
- **nv_mud** : number of Constitutive susbtances type MUD (only if cppkey MUSTANG)
- **nv_sorb** : number of particulate susbtances sorbed on an other particule

---

Note: If MUSTANG cpp key is not defined, **nv_grav**, **nv_sand** and **nv_mud** are not read and **must not be present**

&nmlgravels namelist: each parameter is a vector of length nv_grav

- **name_var_n()** : name of variable
- **long_name_var_n()** : long name of variable
- **standard_name_var_n()** : standard name of variable
- **unit_var_n()** : string, unit of concentration of variable
- **flx_atm_n()** : uniform atmospherical deposition (unit/m2/s)
- **cv_rain_n()** : concentration in rainwater (kg/m3 of water)
- **cini_wat_n()** : initial concentration in water column (kg/m3)
- **cini_air_n()** : initial concentration in air
- **l_out_subs_n()** : saving in output file if TRUE
- **init_cv_name_n()** : name of substance read from initial condition file

- **obc_cv_name_n()** : name of substance read from obc file
- **cini_sed_n()** : initial fraction in the seafloor (only if cppkey MUSTANG)
- **tocd_n()** : critical stress of deposition (N/m2) (only if cppkey MUSTANG)
- **ros_n()** : density of particle (kg/m3) (only if cppkey MUSTANG)
- **l_bedload_n()** : allow bedload transport if true (only if cppkey MUSTANG and key_MUSTANG_V2 and key_MUSTANG_bedload)
- **diam_n()** : diameter of particles (m)

Note: If nv_grav = 0 this namelist is not read.

&nmlsands namelist: each parameter is a vector of length nv_sand

- **name_var_n()** : name of variable
- **long_name_var_n()** : long name of variable
- **standard_name_var_n()** : standard name of variable
- **unit_var_n()** : string, unit of concentration of variable
- **flx_atm_n()** : uniform atmospherical deposition (unit/m2/s)
- **cv_rain_n()** : concentration in rainwater (kg/m3 of water)
- **cini_wat_n()** : initial concentration in water column (kg/m3)
- **cini_air_n()** : initial concentration in air
- **l_out_subs_n()** : saving in output file if TRUE
- **init_cv_name_n()** : name of substance read from initial condition file
- **obc_cv_name_n()** : name of substance read from obc file
- **cini_sed_n()** : initial fraction in the seafloor (only if cppkey MUSTANG)
- **tocd_n()** : critical stress of deposition (N/m2) (only if cppkey MUSTANG)
- **ros_n()** : density of particle (kg/m3) (only if cppkey MUSTANG)
- **l_bedload_n()** : allow bedload transport if true (only if cppkey MUSTANG and key_MUSTANG_V2 and key_MUSTANG_bedload)
- **diam_n()** : diameter of particles (m)
- **l_sand2D()** : treat sand variable as 2D variable if true (used only if key_sand2D, see *Treatment of high settling velocities : SAND variables*)
- **l_outsandrouse()** : if true, use a reconstitution of a ROUSE profil for output in water column (used only if key_sand2D and l_sand2D is TRUE for this variable, see *Treatment of high settling velocities : SAND variables*)

Note: If nv_sand = 0 this namelist is not read.

> **Warning:** If you have several sands in your simulation : start with the coarser sands and continue more and more finely

&nmlmuds namelist: each parameter is a vector of length nv_mud

- **name_var_n()** : name of variable
- **long_name_var_n()** : long name of variable

- **standard_name_var_n()** : standard name of variable
- **unit_var_n()** : string, unit of concentration of variable
- **flx_atm_n()** : uniform atmospherical deposition (unit/m2/s)
- **cv_rain_n()** : concentration in rainwater (kg/m3 of water)
- **cini_wat_n()** : initial concentration in water column (kg/m3)
- **cini_air_n()** : initial concentration in air
- **l_out_subs_n()** : saving in output file if TRUE
- **init_cv_name_n()** : name of substance read from initial condition file
- **obc_cv_name_n()** : name of substance read from obc file
- **cini_sed_n()** : initial fraction in the seafloor (only if cppkey MUSTANG)
- **tocd_n()** : critical stress of deposition (N/m2) (only if cppkey MUSTANG)
- **ros_n()** : density of particle (kg/m3) (only if cppkey MUSTANG)
- **cobc_wat_n()** : boundaries uniform and constant concentration (kg/m3)
- **ws_free_opt_n()** : integer, choice of free settling formulation : 0 constant, 1 Van Leussen, 2 Winterwerp, 3 Wolanski (see *Settling velocity*)
- **ws_free_min_n()** : minimum setling velocity (m/s) (see *Settling velocity*)
- **ws_free_max_n()** : maximum setling velocity (m/s) (see *Settling velocity*)
- **ws_free_para_n(1:4,num substance)** : 4 additional parameters (see *Settling velocity*)
- **ws_hind_opt_n()** : choice of hindered settling formulation : 0 no hindered settling, 1 Scott, 2 Winterwerp, 3 Wolanski (see *Settling velocity*)
- **ws_hind_para_n(1:2,num substance)** : 2 additional parameters (see *Settling velocity*)
- **diam_n()** : diameter of particles (m) (used only if #key_mustang_flocmod is activated see *FLOCMOD*)

Note: If nv_mud = 0 this namelist is not read.

&nmlpartnc namelist: each parameter is a vector of length nv_ncp

- **name_var_n()** : name of variable
- **long_name_var_n()** : long name of variable
- **standard_name_var_n()** : standard name of variable
- **unit_var_n()** : string, unit of concentration of variable
- **flx_atm_n()** : uniform atmospherical deposition (unit/m2/s)
- **cv_rain_n()** : concentration in rainwater (kg/m3 of water)
- **cini_wat_n()** : initial concentration in water column (kg/m3)
- **cini_air_n()** : initial concentration in air
- **l_out_subs_n()** : saving in output file if TRUE
- **init_cv_name_n()** : name of substance read from initial condition file
- **obc_cv_name_n()** : name of substance read from obc file
- **cini_sed_n()** : initial concentration in sediment (quantity/kg of dry sediment) (only if cppkey MUSTANG)
- **tocd_n()** : critical stress of deposition (N/m2) (only if cppkey MUSTANG)

- **ros_n()** : density of particle (kg/m3) (only if cppkey MUSTANG)
- **cobc_wat_n()** : boundaries uniform and constant concentration (kg/m3)
- **ws_free_opt_n()** : integer, choice of free settling formulation : 0 constant, 1 Van Leussen, 2 Winterwerp, 3 Wolanski (see *Settling velocity*)
- **ws_free_min_n()** : minimum setling velocity (m/s) (see *Settling velocity*)
- **ws_free_max_n()** : maximum setling velocity (m/s) (see *Settling velocity*)
- **ws_free_para_n(1:4,num substance)** : 4 additional parameters (see *Settling velocity*)
- **ws_hind_opt_n()** : choice of hindered settling formulation : 0 no hindered settling, 1 Scott, 2 Winterwerp, 3 Wolanski (see *Settling velocity*)
- **ws_hind_para_n(1:2,num substance)** : 2 additional parameters (see *Settling velocity*)

Note: If nv_ncp = 0 this namelist is not read.

If MUSTANG cpp key is not defined, **cini_sed_n**, **tocd_n**, **ros_n**, **ws_free_opt_n**, **ws_free_para_n**, **ws_hind_opt_n**, **ws_hind_para_n** are not read and must not be present

---

&nmlpartsorb namelist: each parameter is a vector of length nv_sorb

- **name_var_n()** : name of variable
- **long_name_var_n()** : long name of variable
- **standard_name_var_n()** : standard name of variable
- **unit_var_n()** : string, unit of concentration of variable
- **flx_atm_n()** : uniform atmospherical deposition (unit/m2/s)
- **cv_rain_n()** : concentration in rainwater (kg/m3 of water)
- **cini_wat_n()** : initial concentration in water column (kg/m3)
- **cini_air_n()** : initial concentration in air
- **l_out_subs_n()** : saving in output file if TRUE
- **init_cv_name_n()** : name of substance read from initial condition file
- **obc_cv_name_n()** : name of substance read from obc file
- **cini_sed_n()** : initial concentration in sediment (quantity/kg of dry sediment) (only if cppkey MUSTANG)
- **cobc_wat_n()** : boundaries uniform and constant concentration (kg/m3)
- **name_varpc_assoc_n()** : name of associated particulate substance on which this substance is sorbed

---

Note: If nv_sorb = 0 this namelist is not read.

If MUSTANG cpp key is not defined, **cini_sed_n** is not read and must not be present

---

&nmlvardiss namelist: each parameter is a vector of length nv_dis

- **name_var_n()** : name of variable
- **long_name_var_n()** : long name of variable
- **standard_name_var_n()** : standard name of variable
- **unit_var_n()** : string, unit of concentration of variable
- **flx_atm_n()** : uniform atmospherical deposition (unit/m2/s)

- **cv_rain_n()** : concentration in rainwater (kg/m3 of water)
- **cini_wat_n()** : initial concentration in water column (kg/m3)
- **cini_air_n()** : initial concentration in air
- **l_out_subs_n()** : saving in output file if TRUE
- **init_cv_name_n()** : name of substance read from initial condition file
- **obc_cv_name_n()** : name of substance read from obc file
- **cini_sed_n()** : initial concentration in sediment (quantity/kg of dry sediment) (only if cppkey MUSTANG)
- **cobc_wat_n()** : boundaries uniform and constant concentration (kg/m3)

Note: If nv_dis = 0 this namelist is not read.

If MUSTANG cpp key is not defined, **cini_sed_n** is not read and must not be present

---

&nmlvarfix namelist: each parameter is a vector of length nv_fix

- **name_var_fix()** : name of variable
- **long_name_var_fix()** : long name of variable
- **standard_name_var_fix()** : standard name of variable
- **unit_var_fix()** : string, unit of concentration of variable
- **cini_wat_fix()** : initial concentration in water column (kg/m3)
- **l_out_subs_fix()** : saving in output file if TRUE
- **init_cv_name_fix()** : name of substance read from initial condition file

---

Note: If nv_fix = 0 this namelist is not read.

&nmlvarbent namelist: each parameter is a vector of length nv_bent

- **name_var_bent()** : name of variable
- **long_name_var_bent()** : long name of variable
- **standard_name_var_bent()** : standard name of variable
- **unit_var_bent()** : string, unit of concentration of variable
- **cini_bent()** : initial concentration
- **l_out_subs_bent()** : saving in output file if TRUE

Note: If nv_bent = 0 or cppkey key_benthic is not defined, this namelist is not read.

Input file : Mustang namelist

Note: If the user does not specify a parameter in the namelist file, the default value in MUSTANG_NAMELIST/paraMUSTANG_default.txt is used

This default input files incorporates all the parameters that are needed for both V1 or V2 MUSTANG versions. Therefore, not all parameters are used, depending on the set of CPP keys or booleans included in the MUSTANG input file itself.

All the parameters in the paraMUSTANG_default.txt are read first, before reading the user-defined input file defined in croco.in. The parameters defined in the user-defined namelist file will overwrite those defined in the default file. The user-defined input file can either be a full copy of the default input file, or only define the parameters that matter the most for a specific configuration. In the later case, even if the parameters are not mentionned, the namelist group section needs to be present in the file even if empty, e.g.:

```
&namsedim_deposition
/
```

Mustang input file contains at least 10 namelists and at most 16 namelists depending on the cpp-keys key_MUSTANG_V2, key_MUSTANG_debug, key_MUSTANG_flocmod key_MUSTANG_lateralerosion, key_noTSdiss_insed :

- *namsedim_init* : relative to sediment initialization
- *namsedim_layer* : relative to sediment layers characterization and active layer
- *namsedim_bottomstress* : relative to bottom shear stress
- *namsedim_deposition* : relative to sediment deposition
- *namsedim_erosion* : relative to sediment erosion
- *namsedim_poro* : relative to porosity (only if key_MUSTANG_V2)
- *namsedim_bedload* : relative to sediment bedload (only if key_MUSTANG_V2)
- *namsedim_lateral_erosion* : relative to lateral sediment erosion (only if key_MUSTANG_lateralerosion)
- *namsedim_consolidation* : relative to sediment consolidation
- *namsedim_diffusion* : relative to dissolved diffusion in sediment
- *namsedim_bioturb* : relative to bioturbation in sediment
- *namsedim_morpho* : relative to morphodynamic
- *namtempsed* : relative to temperature estimation in sediment (only if !defined key_noTSdiss_insed)
- *namsedoutput* : parameters used for output results in the file sediment
- *namsedim_debug* : output for debug (only if key_MUSTANG_debug and key_MUSTANG_V2)
- *namflocmod* : parameters using for FLOCMOD module (only if key_MUSTANG_flocmod)

Each namelist is described bellow with the description of each parameter.

&namsedim_init :

- **date_start_dyninsed** : starting date for dynamic processes in sediment
- **date_start_morpho** : starting date for morphodynamic processes
- **l_repsed** : set to .true. if sedimentary variables are initialized from a previous run
- **filrepsed** : file path from which the model is initialized for the continuation of a previous run. WARNING : filrepsed must be given if l_bathy_actu = .T. in order to read new h0 even if l_repsed = .F.

- **l_initsed_vardiss** : set to .true. if initialization of dissolved variables, temperature and salinity in sediment (will be done with concentrations in water at bottom (k=1))
- **l_unised** : set to .true. for a uniform bottom initialization
- **fileinised** : file path for initialization (if l_unised is False)
- **hseduni** : initial uniform sediment thickness (m)
- **cseduni** : initial sediment concentration (kg/m3)
- **l_init_hsed** : set to .true. if we want to adjust the sediment thickness in order to be coherent with sediment parameters (calculation of a new hseduni based on cseduni, cvolmax values, and csed_ini of each sediment)
- **csed_mud_ini** : mud concentration into initial sediment (kg/m3) (if = 0. ==> csed_mud_ini = cfreshmud)
- **ksmiuni** : lower grid cell index in the sediment
- **ksmauni** : upper grid cell index in the sediment
- **sini_sed** : initial interstitial water uniform salinity (in the sediment) (PSU)
- **tini_sed** : initial interstitial water uniform temperature (in the sediment) (Celsius degree)
- **poro_mud_ini** : if key_MUSTANG_V2 only, initial porosity of mud fraction

&namsedim_layer :

- **l_dzsminuni** : set to .false. if dzsmin vary with sediment bed composition, else dzsmin = dzsminuni (used if key_MUSTANG_V2 only)
- **dzsminuni** : minimum sediment layer thickness (m) (used if key_MUSTANG_V2 only)
- **dzsmin** : minimum sediment layer thickness (m)
- **dzsmax_bottom** : maximum thickness of bottom layers which result from the fusion when ksdmax is exceeded (m)
- **l_dzsmaxuni** : if set to .true. dzsmax = dzsmaxuni , if set to .false. then linearly computed in MUSTANG_sedinit from dzsmaxuni to dzsmaxuni/100 depending on water depth
- **dzsmaxuni** : uniform maximum thickness for the superficial sediment layer (m), must be >0
- **nlayer_surf_sed** : number of layers below the sediment surface that can not be melted (max thickness = dzsmax)
- **k1HW97** : ref value k1HW97 = 0.07, parameter to compute active layer thickness (Harris and Wiberg, 1997) (key_MUSTANG_V2 only)
- **k2HW97** : ref value k2HW97 = 6.0, parameter to compute active layer thickness (Harris and Wiberg, 1997) (key_MUSTANG_V2 only)
- **fusion_para_activlayer** : criterion cohesiveness for fusion in active layer (key_MUSTANG_V2 only) :
  - 0 : no fusion,
  - = 1 : frmudcr1,
  - > 1 : between frmudcr1 & frmudcr2

&namsedim_bottomstress, this part combine *bottom stress* and *roughness* parameters :

- **l_z0seduni** : if true, z0seduni is used; if false z0sed is computed from sediment diameter
- **z0seduni** : uniform bed roughness (m)
- **z0sedmud** : mud (i.e.minimum) bed roughness (m) (used only if l_unised is false)
- **z0sedbedrock** : bed roughness for bedrock (no sediment) (m) (used only if l_unised is false)

- **l_fricwave** : if true the wave related friction factor is computed from wave orbital velocity and period; if false then fricwav namelist value is used (see *wave skin friction*)
- **fricwav** : default value is 0.06, wave related friction factor (used for bottom shear stress computation if l_fricwave is false)
- **l_z0hydro_coupl_init** : if true the evaluation of z0 hydro depends on sediment composition at the beginning of the simulation
- **l_z0hydro_coupl** : if true the evaluation of z0 hydro depends on sediment composition along the run
- **coef_z0_coupl** : if l_z0hydro_coupl is true, parameter to compute z0hydro in the first centimeter z0hydro = coef_z0_coupl * sand diameter
- **z0_hydro_mud** : if l_z0hydro_coupl is true, z0hydro if pure mud (m)
- **z0_hydro_bed** : if l_z0hydro_coupl is true, z0hydro if no sediment (m)

&namsedim_deposition :

- **cfreshmud** : prescribed fresh deposit concentration in kg/m3 (must be around 100 if consolidation or higher (300-500 if no consolidation)
- **csedmin** : concentration of the upper layer under which there is fusion with the underlying sediment cell (in kg/m3)
- **cmudcr** : critical relative concentration of the surface layer above which no mixing is allowed with the underlying sediment (in kg/m3)
- **aref_sand** : reference height above sediment in meter. Used for computing of sand deposit for sand extrapolation on water column and correct sand transport, value by default = 0.02 correspond to Van Rijn experiments. **DO NOT CHANGED IF NOT EXPERT**. (see *Treatment of high settling velocities : SAND variables*)
- **cvolmaxsort** : maximum volumic concentration of sorted sand
- **cvolmaxmel** : maximum volumic concentration of mixed sediments
- **slopefac** : slope effect multiplicative on sliding part of deposit (only if key_MUSTANG_slipdeposit see *sliding fluxes*)

&namsedim_erosion :

- **activlayer** : active layer thickness (m)
- **frmudcr2** : critical mud fraction under which the behaviour is purely sandy
- **coef_frmudcr1** : such that critical mud fraction under which sandy behaviour (frmudcr1=min(coef_frmudcr1*d50 sand,frmudcr2))
- **x1toce_mud** : mud erosion parameter : toce = x1_toce_mud*(relative mud concentration)**x2_toce_mud
- **x2toce_mud** : mud erosion parameter: toce = x1_toce_mud*(relative mud concentration)**x2_toce_mud
- **E0_sand_option** : choice of formulation for E0_sand evaluation :
  - 0 : E0_sand = E0_sand_Cst read in this namelist
  - 1 : E0_sand evaluated with Van Rijn (1984) formulation
  - 2 : E0_sand evaluated with erodimetry (min(0.27,1000*d50-0.01)*toce**n_eros_sand)
  - 3 : E0_sand evaluated with Wu and Lin (2014) formulation
- **E0_sand_Cst** : constant erosion flux for sand (used if E0_sand_option= 0)
- **E0_sand_para** : coefficient used to modulate erosion flux for sand (=1 if no correction)

- **n_eros_sand** : parameter for erosion flux for sand (E0_sand*(tenfo/toce-1.)**n_eros_sand). WARNING : choose parameters compatible with E0_sand_option (example : n_eros_sand=1.6 for E0_sand_option=1)
- **E0_mud** : parameters for erosion flux for pure mud
- **E0_mud_para_indep** : parameter to correct E0_mud in case of erosion class by class in non cohesive regime (key_MUSTANG_V2 only)
- **n_eros_mud** : E0_mud*(tenfo/toce-1.)**n_eros_mud
- **ero_option** : choice of erosion formulation for mixing sand-mud. **These formulations are debatable and must be considered carefully by the user. Other laws are possible and could be programmed.**
  - 0 : pure mud behavior (for all particles and whatever the mixture)
  - 1 : linear interpolation between sand and mud behavior, depend on proportions of the mixture
  - 2 : formulation derived from that of J.Vareilles (2013)
  - 3 : formulations proposed by B. Mengual (2015) with exponential coefficients depend on proportions of the mixture
- **l_xexp_ero_cst** : set to .true. if xexp_ero estimated from empirical formulation, depending on frmudcr1 (key_MUSTANG_V2 only)
- **xexp_ero** : used only if ero_option=3 : adjustment on exponential variation (more brutal when xexp_ero high)
- **tau_cri_option** : ichoice of critical stress formulation
  - 0: Shields
  - 1: Wu and Lin (2014)
- **tau_cri_mud_option_eroindep** : choice of mud critical stress formulation
  - 0: x1toce_mud*cmudr**x2toce_mud
  - 1: toce_meansan if somsan>eps (else->case0)
  - 2: minval(toce_sand*cvsed/cvsed+eps) if >0 (else->case0)
  - 3: min(case 0; toce(isand2)) (key_MUSTANG_V2 only)
- **l_eroindep_noncoh** :
  - set to .true. in order to activate independant erosion for the different sediment classes sands and muds
  - set to .false. to have the mixture mud/sand eroded as in version V1 (key_MUSTANG_V2 only)
- **l_eroindep_mud** :
  - set to .true. if mud erosion independant for sands erosion
  - set to .false. if mud erosion proportionnal to total sand erosion (key_MUSTANG_V2 only)
- **l_peph_suspension**: set to .true. if hindering / exposure processes in critical shear stress estimate for suspension (key_MUSTANG_V2 only)

&namsedim_poro : (key_MUSTANG_V2 only)

- **poro_option** : choice of porosity formulation
  - 1: Wu and Li (2017) (incompatible with consolidation))
  - 2: mix ideal coarse/fine packing
- **poro_min** : minimum porosity below which consolidation is stopped

- **Awooster** : parameter of the formulation of Wooster et al. (2008) for estimating porosity associated to the non-cohesive sediment see Cui et al. (1996); ref value = 0.42
- **Bwooster** : parameter of the formulation of Wooster et al. (2008) for estimating porosity associated to the non-cohesive sediment see Cui et al. (1996); ref value = -0,458
- **Bmax_wu** : maximum portion of the coarse sediment class participating in filling; ref value = 0.65

&namsedim_bedload : (key_MUSTANG_V2 only)

- **l_peph_bedload** : set to .true. if hindering / exposure processes in critical shear stress estimate for bedload
- **l_slope_effect_bedload** : set to .true. if accounting for slope effects in bedload fluxes (Lesser formulation)
- **alphabs** : coefficient for slope effects (default coefficients Lesser et al. (2004), alphabs = 1.)
- **alphabn** : coefficient for slope effects (default coefficients Lesser et al. (2004), default alphabn is 1.5 but can be higher, until 5-10 (Gerald Herling experience))
- **hmin_bedload** : no bedload in u/v directions if h0+ssh <= hmin_bedload in neighbouring cells
- **l_fsusp** : limitation erosion fluxes of non-coh sediment in case of simultaneous bedload transport, according to Wu & Lin formulations. Set to .true. if erosion flux is fitted to total transport should be set to .false. if E0_sand_option=3 (Wu & Lin)

&namsedim_lateral_erosion : (see *Lateral erosion*)

- **coef_erolat** : slope effect multiplicative factor
- **coef_tauskin_lat** : parameter to evaluate the lateral stress as a function of the average tangential velocity on the vertical
- **l_erolat_wet_cell** : set to .true in order to take into account wet cells lateral erosion
- **htncrit_eros** : critical water height so as to prevent erosion under a given threshold (the threshold value is different for flooding or ebbing, cf. Hibma's PhD, 2004, page 78)

&namsedim_consolidation :

- **l_consolid** : set to .true. if sediment consolidation is accounted for
- **dt_consolid** : time step for consolidation processes in sediment (will use in fact the min between dt_consolid, dt_diffused if l_diffused, dt_bioturb if l_bioturb)
- **subdt_consol** : sub time step for consolidation processes in sediment (< or = min(dt_consolid, ..))(will use in fact the min between subdt_consolid, subdt_bioturb if l_bioturb)
- **csegreg** : DO NOT CHANGE VALUE if not expert, default 250.0
- **csandseg** : DO NOT CHANGE VALUE if not expert, default 1250.0
- **xperm1** : parameter to compute permeability permeability=xperm1*d50*d50*voidratio**xperm2
- **xperm2** : parameter to compute permeability permeability=xperm1*d50*d50*voidratio**xperm2
- **xsigma1** : parameter used in Merckelback & Kranenburg s (2004) formulation. DO NOT CHANGE VALUE if not expert, default 6.0e+05
- **xsigma2** : real, parameter used in Merckelback & Kranenburg s (2004) formulation. DO NOT CHANGE VALUE if not expert, default 6

&namsedim_diffusion :

- **l_diffused** : set to .true. if taking into account dissolved diffusion in sediment and at the water/sediment interface

- **dt_diffused** : time step for diffusion processes in sediment (will use in fact the min between dt_diffused, dt_consolid if l_consolid, dt_bioturb if l_bioturb)
- **choice_flxdiss_diffsed** : choice for expression of dissolved fluxes at sediment-water interface
  - 1 : Fick law : gradient between Cv_wat at dz(1)/2
  - 2 : Fick law : gradient between Cv_wat at distance epdifi
- **xdifs1** : diffusion coefficients within the sediment
- **xdifsi1** : diffusion coefficients at the water sediment interface
- **epdifi** : diffusion thickness in the water at the sediment-water interface
- **fexcs** : factor of eccentricity of concentrations in vertical fluxes evaluation (.5 a 1) (numerical scheme for dissolved diffusion/advection(by consol) in sediment)

&namsedim_bioturb :

- **l_bioturb** : set to .true. if taking into account particulate bioturbation (diffusive mixing) in sediment
- **l_biodiffs** : set to .true. if taking into account dissolved bioturbation diffusion in sediment
- **dt_bioturb** : time step for bioturbation processes in sediment (will use in fact the min between dt_bioturb, dt_consolid if l_consolid, dt_diffused if l_diffused)
- **subdt_bioturb** : sub time step for bioturbation processes in sediment (< or = min(dt_bioturb, ..)) (will use in fact the min between subdt_bioturb, subdt_consolid if l_consolid)
- **xbioturbmax_part** : max particular bioturbation coefficient by bioturbation Db (in surface)
- **xbioturbk_part** : coef (slope) for part. bioturbation coefficient between max Db at sediment surface and 0 at bottom
- **dbiotu0_part** : max depth beneath the sediment surface below which there is no bioturbation
- **dbiotum_part** : sediment thickness where the part-bioturbation coefficient Db is constant (max)
- **xbioturbmax_diss** : max diffusion coeffient by biodiffusion Db (in surface)
- **xbioturbk_diss** : coef (slope) for biodiffusion coefficient between max Db at sediment surface and 0 at bottom
- **dbiotu0_diss** : max depth beneath the sediment surface below which there is no bioturbation
- **dbiotum_diss** : sediment thickness where the diffsolved-bioturbation coefficient Db is constant (max)
- **frmud_db_min** : mud fraction limit (min) below which there is no Biodiffusion
- **frmud_db_max** : mud fraction limit (max)above which the biodiffusion coefficient Db is maximum (muddy sediment)

&namsedim_morpho :

- **l_morphocoupl** : set to .true if coupling module morphodynamic, see *Morphodynamic*
- **MF** : morphological factor : multiplication factor for morphological evolutions, equivalent to a "time acceleration" (morphological evolutions over a MF*T duration are assumed to be equal to MF * the morphological evolutions over T).
- **dt_morpho** : time step for morphodynamic (s)
- **l_MF_dhsed** :
  - set to .true. if morphodynamic applied with sediment height variation amplification
  - set to .false. if morphodynamic is applied with erosion/deposit fluxes amplification
- **l_bathy_actu** : set to .true. if reading a new bathy issued a previous run and saved in filrepsed (given in namelist namsedim_init) **!!! NOT IMPLEMENTED YET !!!**

&namtempsed : (only if !defined key_noTSdiss_insed)

- **mu_tempsed1** : parameters used to estimate thermic diffusitiy function of mud fraction
- **mu_tempsed2** : parameters used to estimate thermic diffusitiy function of mud fraction
- **mu_tempsed3** : parameters used to estimate thermic diffusitiy function of mud fraction
- **epsedmin_tempsed** : sediment thickness limits for estimation heat loss at bottom, if hsed < epsedmin_tempsed : heat loss at sediment bottom = heat flux a sediment surface
- **epsedmax_tempsed** : sediment thickness limits for estimation heat loss at bottom, if hsed > epsedmax_tempsed : heat loss at sediment bottom = 0.

&namsedoutput :

- **l_outsed_saltemp** : set to .true. if output Salinity and Temperature in sediment
- **l_outsed_flx_WS_all** : set to .true. if output fluxes threw interface Water/sediment (2 2D variables per constitutive particulate variable)
- **l_outsed_flx_WS_int** : set to .true. if output fluxes threw interface Water/sediment (integration on all constitutive particulate variables)
- **choice_nivsed_out** : choice of saving output
  - 1 : all the layers (ksdmin to ksdmax) are saved (k=1 : bottom layer) (nk_nivsed_out, ep_nivsed_out, epmax_nivsed_out are not used)
  - 2 : only save the nk_nivsed_out surficial layers (k=1 : layer most bottom)
  - 3 : each layers from sediment surface are saved till the thickness epmax_nivsed_out (which must be non zero and > dzsmax (k=1 : bottom layer)) This option is not recommended if l_dzsmaxuni=.False.
  - 4 : 1 to 5 layers of constant thickness are saved; thickness are selected with ep_nivsed_out and concentrations are interpolated to describe the sediment thickness (k=1 : surface layer) the thickness of the bottom layer (nk_nivsed_out+1) will vary depending on the total thickness of sediment in the cell
- **nk_nivsed_out** : number of saved sediment layers
  - unused if choice_nivsed_out = 1
  - <ksdmax if choice_nivsed_out = 2,
  - unused if choice_nivsed_out = 3
  - <6 if choice_nivsed_out = 4,
- **ep_nivsed_out()** : 5 values of sediment layer thickness (mm), beginning with surface layer (used if choice_nivsed_out=4)
- **epmax_nivsed_out** : maximum thickness (mm) for output each layers of sediment (used if choice_nivsed_out=3). Below the layer which bottom level exceed this thickness, an addition layer is an integrative layer till bottom

&namsedim_debug :

- **l_debug_effdep** : set to .true. if print some informations for debugging MUSTANG deposition
- **l_debug_erosion** : set to .true. if print informations for debugging in erosion routines
- **date_start_debug** : string, starting date for write debugging informations
- **lon_debug** : define mesh location where we print these informations
- **lat_debug** : define mesh location where we print these informations
- **i_MUSTANG_debug** : indexes of the mesh where we print these informations (only if lon_debug and lat_debug = 0.)

- **j_MUSTANG_debug** : indexes of the mesh where we print these informations (only if lon_debug and lat_debug = 0.)

&namflocmod :

- **l_ADS** : set to .true. if aggregation by differential settling
- **l_ASH** : set to .true. if aggregation by shear
- **l_COLLFRAG** : set to .true. if fragmentation by collision
- **f_dp0** : primary particle size (default 4.e-6 m)
- **f_nf** : fractal dimension (default 2.0, usual range from 1.6 to 2.8)
- **f_nb_frag** : nb of fragments of equal size by shear fragmentation (default 2.0 as binary fragmentation)
- **f_alpha** : flocculation efficiency parameter (default 0.15)
- **f_beta** : floc break up parameter (default 0.1)
- **f_ater** : ternary fragmentation factor : proportion of flocs fragmented as half the size of the initial binary fragments (0.0 if full binary fragmentation, 0.5 if ternary fragmentation)
- **f_ero_frac** : floc erosion (% of floc mass eroded) (default 0.05)
- **f_ero_nbfrag** : nb of fragments produced by erosion (default 2.0)
- **f_ero_iv** : fragment class (mud variable index corresponding to the eroded particle size - typically 1)
- **f_mneg_param**: negative mass after flocculation/fragmentation allowed before redistribution (default 0.001 g/l)
- **f_dmin_frag** : minimum diameter for fragmentation (default 10e-6 microns)
- **f_cfcst** : fraction of mass lost by flocs if fragmentation by collision .. (default : =3._rsh/16._rsh)
- **f_fp** : relative depth of inter particle penetration (default =0.1) (McAnally, 1999)
- **f_fy** : floc yield strength (default= 1.0e-10) (Winterwerp, 2002)
- **f_collfragparam** : fraction of shear aggregation leading to fragmentation by collision (default 0.0, must be less than 1.0)

Input file : Initialization of the sediment cover

If the initialization of sediment bed is not uniform (spatially and vertically), the sediment cover is read from a netcdf file (see *Initialization* for more details).

This netcdf file path is given in *MUSTANG namelist &namsedim_init* as follow:

```
l_repsed=.true.
filrepsed='./repsed.nc'
```

The netcdf file needs to have the concentration values under the names *NAME_sed*, with NAME corresponding to the names defined in the *SUBSTANCE namelist*. The number of vertical levels (ksmi, ksma) and the layer thickness (DZS) also need to be defined. The file structure is similar to the RESTART netcdf file, and filerepsed can be used to restart from a CROCO RESTART file (same format).

Header of an example sediment cover file:

```
dimensions:
     ni = 821 ;
     nj = 623 ;
     time = UNLIMITED ; // (1 currently)
          level = 10 ;
```

(continues on next page)

(continued from previous page)

```
         variables:
     double latitude(nj, ni) ;
     double longitude(nj, ni) ;
     double time(time) ;
     double level(level) ;
     double ksmi(time, nj, ni) ;
     double ksma(time, nj, ni) ;
     double DZS(time, level, nj, ni) ;
     double temp_sed(time, level, nj, ni) ;
     double salt_sed(time, level, nj, ni) ;
     double GRAV_sed(time, level, nj, ni) ;
     double SAND_sed(time, level, nj, ni) ;
     double MUD1_sed(time, level, nj, ni) ;
```

Input file : Wave

If cpp keys **WAVE_OFFLINE** and **MUSTANG** are activated, a netcdf file must be given in CROCO input file **croco.in** as follow:

```
wave_offline:   filename
      wave.nc
```

Significant wave height, wave period, wave direction and bottom orbital velocity are read in this netcdf file. Note that the significant wave height (or wave amplitude) has to be given as for now but is not used to compute the bed shear stress.

The netcdf file should have a temporal window at leats covering the simulation period. Temporal interpolation are made between each time step in the file. **If your configuration contains wetting-drying effect, be careful with values on land, do not put negative values**. We advice you to put 0. for hs, dir and ubr and 10 for t01 (to avoid division by zero).

Header of an example wave file:

```
dimensions:
    wwv_time = UNLIMITED ; // (25 currently)
    eta_rho = 132 ;
    xi_rho = 182 ;
variables:
    double dir(wwv_time, eta_rho, xi_rho) ;
        dir:_FillValue = 0s ;
    double hs(wwv_time, eta_rho, xi_rho) ;
        hs:_FillValue = 0s ;
    double t01(wwv_time, eta_rho, xi_rho) ;
        t01:_FillValue = 10s ;
    double ubr(wwv_time, eta_rho, xi_rho) ;
        ubr:_FillValue = 0s ;
    double uubr(wwv_time, eta_rho, xi_rho) ;
        uubr:_FillValue = 0s ;
    double vubr(wwv_time, eta_rho, xi_rho) ;
        vubr:_FillValue = 0s ;
    double wwv_time(wwv_time) ;
```

Input file : Solid discharge in rivers

If cpp keys **PSOURCE_NCFILE** and **PSOURCE_NCFILE_TS** are activated, a netcdf file must be given in CROCO input file **croco.in** as follow:

```
psource_ncfile:   Nsrc  Isrc  Jsrc  Dsrc qbardir  Lsrc  Tsrc   runoff file name
        psource.nc
    2
                   167  56  0 -1   30*T   20.0  15.0   Loire
                   91  99  0 -1   30*T   20.0  15.0   Vilaine_arzal
```

This file is in netcdf format. It reads the concentration values (T,S and sediment) in get_psource_ts.F.

The name of sediment variable must match the name chosen in *SUBSTANCE namelist*, in the example below : MUD1.

Header of an example source file:

```
dimensions:
    qbar_time = 7676 ;
    n_qbar = 6 ;
    runoffname_StrLen = 30 ;
    temp_src_time = 8037 ;
    salt_src_time = 8037 ;
    MUD1_src_time = 7676 ;
        variables:
    double qbar_time(qbar_time) ;
        qbar_time:long_name = "runoff time" ;
        qbar_time:units = "days" ;
        qbar_time:cycle_length = 0 ;
        qbar_time:long_units = "days since 1900-01-01" ;
    double Qbar(n_qbar, qbar_time) ;
        Qbar:long_name = "runoff discharge" ;
        Qbar:units = "m3.s-1" ;
    char runoff_name(n_qbar, runoffname_StrLen) ;
    double temp_src_time(temp_src_time) ;
        temp_src_time:cycle_length = 0 ;
        temp_src_time:long_units = "days since 1900-01-01" ;
    double salt_src_time(salt_src_time) ;
        salt_src_time:cycle_length = 0 ;
        salt_src_time:long_units = "days since 1900-01-01" ;
    double temp_src(n_qbar, temp_src_time) ;
        temp_src:long_name = "runoff temperature" ;
        temp_src:units = "Degrees Celcius" ;
    double salt_src(n_qbar, salt_src_time) ;
        salt_src:long_name = "runoff salinity" ;
        salt_src:units = "psu" ;
    double MUD1_src_time(MUD1_src_time) ;
        MUD1_src_time:long_name = "runoff time" ;
        MUD1_src_time:units = "days" ;
        MUD1_src_time:long_units = "days since 1900-01-01" ;
    double MUD1_src(n_qbar, MUD1_src_time) ;
```

Output options

Outputs for sediment variables are written by CROCO not MUSTANG, using routines that have been modified on purpose (e.g. wrt_his.F):

- Classic Netcdf outputs with suspended sediment concentrations and various variables within the sediment bed (NB_LAY_SED, HSED, thickness DZS, shear stress TAUSKIN, concentration in sediment bed for each sediment (*_sed), temperature and salinity in sediment bed (temp_sed and salt_sed)). Use **#key_MUSTANG_specif_outputs** to add more sediment variables in this file. NB : For now, it is not possible to select only a few variable for output.
- Station files (#define STATIONS) only record suspended sediment concentrations. Sediment bed variables are not implemeted yet.
- XIOS (#define XIOS) can be configured to output both suspended sediment concentrations and sediment bed variables.

Available CPP keys

The compulsory CPP keys to use MUSTANG in CROCO:

- **MUSTANG** : activate module MUSTANG
- **SUBSTANCE** : activate module SUBSTANCE
- **SALINITY** : needed for SUBSTANCE
- **TEMPERATURE** : needed for SUBSTANCE
- **USE_CALENDAR** : needed for MUSTANG, issue with MUSTANG coupling timing otherwise
- **key_noTSdiss_insed** : temperature, salinity and others dissolved variables are not computed in sediment. They have constant values and no fluxes of dissolved variables between water and sediment are computed.
- **key_nofluxwat_IWS** : no exchange water fluxes between water and sediment (recommended if key_noTSdiss_insed).

The CPP keys, not fully compatible with MUSTANG:

- **FILLVAL** : if defined, this key does not allow to write restart file with MUSTANG

The optional CPP keys, to choose processes or version:

- **key_MUSTANG_V2** : to use MUSTANG in V2, without this key, the version V1 is used
- **MORPHODYN** : to activate morphodynamic (**l_morphocoupl** must also be set to .true, see *Morphodynamic*)
- **key_sand2D** : to treat SAND in suspension as 2D variable, see *Treatment of high settling velocities : SAND variables*
- **MUSTANG_CORFLUX** : to correct SAND horizontal fluxes, see *Treatment of high settling velocities : SAND variables*
- **WAVE_OFFLINE** : to use wave in bed shear stress computation, see *wave skin friction*
- **key_MUSTANG_flocmod** : to activate module floculation (*FLOCMOD*)
- **key_tauskin_c_upwind** : Upwind scheme for current-induced bottom shear stress, see *Shear stress*
- **key_tauskin_c_center** : Compute bottom shear stress with u^* directly at (rho) location (center of the cell), see *Shear stress*
- **key_tauskin_c_ubar** : Shear stress computed form depth-averaged velocity, see *Shear stress*
- **PSOURCE_NCFILE** and **PSOURCE_NCFILE_TS** : to read solid discharge in river from netcdf files
- **key_MUSTANG_slipdeposit** : see *Sliding fluxes*

- **key_MUSTANG_lateralerosion** : see *Lateral erosion*
- **key_MUSTANG_splitlayersurf** : cutting of surface sediment layers to have regular and precise discretization at surface
- **key_MUSTANG_specif_outputs** : output more diagnostics variables
- **key_MUSTANG_bedload** : only with key_MUSTANG_V2, bedload processus included
- **key_MUSTANG_debug** : only with key_MUSTANG_V2, choice print information during erosion or deposit process. **Does not work in MPI** print a lot of variable during run for debugging choice of coordinates of the point to be checked

The following CPP keys are **not yet available with CROCO** :

- key_MUSTANG_add_consol_outputs, only with key_MUSTANG_V2 : outputs more diagnostics variables related to consolidation
- key_MUSTANG_roswat_with_mud : to take into account the influence on water density of high concentrations of particles

MUSTANG technical documentation

Overall architecture of the module

To compute sediment behavior, MUSTANG execute the steps :

- *Initialization* of sediment variables from input files
- *Temporal loop* with a sequence of forcing update, erosion phase, exchange between sediment and water, deposition phase, morphodynamic update and call to output feature

MUSTANG module is integrated into CROCO code. It is composed of 14 elements added to the existing code in a MUSTANG directory. The interface between the sedimentary module and the hydrodynamic code is done via :

- plug_MUSTANG_CROCO.F90 which makes the 4 main subroutines available to the rest of the code :
  - mustang_init_main : initialization
  - mustang_update_main : update of forcing and erosion phase
  - mustang_deposition_main : deposition phase
  - mustang_morpho_main : to apply morphodynamic
- modification of CROCO files :
  - Initialization in main.F
  - Main calls in step.F
  - Treatment of settling and sediment/water exchanges in step3d_t.F
  - Input features in read_inp.F
  - Output features in : XIOS/send_xios_diags.F, OCEAN/wrt_his.F, OCEAN/wrt_rst.F, OCEAN/wrt_sta.F, OCEAN/nc_sta.h, OCEAN/ncscrum.h, OCEAN/nf_read.h, OCEAN/fillvalue.F, step_sta.F, sta.h
  - Wave forcing in OCEAN directory in files : forces.h, get_vbc.F, get_wwave.F, init_arrays.F, init_scalars.F
  - Test cases in OCEAN/ana_grid.F and OCEAN/ana_initial.F
  - Compilation and dimension in OCEAN directory in files : Makefile, jobcomp, param.h, cppdefs.h, cppdefs_dev.h

Roughly, all sediment calculations are done in the MUSTANG fortran files, except for the advection part which is intimately linked to step3d_t.F. Modifications in CROCO files are mostly for I/O features or test cases analytical forcing.

Initialization

The initialization must be done for water column variables and sediment bed variables.

In the water column, the initialization is done from initial file if provided (see hydrodynamic code). If there is no initial file, the water concentrations are initialized from uniform value in *SUBSTANCE namelist*.

To initialize the sediment cover, two options are available :

- Uniform sediment cover

  In paraMUSTANG*.txt:

```
l_unised = .true.  ! boolean set to true for a uniform bottom␣
↪initialization
hseduni = 0.03     ! initial uniform sediment thickness(m)
cseduni= 1500.0    ! initial sediment concentration
csed_mud_ini = 550.0  ! mud concentration into initial sediment (if =␣
↪0. ==> csed_mud_ini = cfreshmud)
ksmiuni = 1        ! lower grid cell indices in the sediment
ksmauni = 10       ! upper grid cell indices in the sediment
```

  And then, the fraction of each sediment variable in the seafloor is defined with *cini_sed_n()* in parasubsance_MUSTANG.txt

- Read the sediment cover from a netcdf file (format of a RESTART file, see *input file for sediment cover*)

> **Warning:**
>
> – The restarts are not *perfect* restarts. To do a perfect restart, you will need to save the erosion and deposition fluxes in a restart file, as was done in MARS3D (cf. subroutine sed_outsaverestart in sed_MUSTANG_CROCO.F90). This has not been implemented yet.
>
> – Further, it will not work for morphological runs as you will need to make a few changes to read the bathymetry from the *filerepsed* file.

Temporal loop

In the temporal loop of CROCO model, the main calls to MUSTANG routines are in step3D_t_thread.

```
call prestep3D_thread()
call step2d_thread()
call step3D_uv_thread()
call step3D_t_thread()
----> call mustang_update_main()
----> call step3d_t
-----------------> #  include "t3dmix_tridiagonal_settling.h"
----> CALL mustang_deposition_main
----> CALL mustang_morpho_main
```

The erosion and deposition phases are sequenced at each time step:

- erosive phase is treated before the call to step3d_t (treatment of vertical advection). This phase contains:
  - calculation of the *roughness* and *bottom shear stress*,
  - calculation of the *erosion fluxes* for each class
  - evolution of the sedimentary bed from erosion : *erosion layer managment*
  - calculation of the tendency to deposition *deposit fluxes*.

- exchanges from the water column point of view are processed in step3d_t via "t3dmix_tridiagonal_settling.h" and compute also the effective deposit fluxes
- deposit step is processed after the call to step3d_t. This phase includes
  - calculation of the deposit for each class,
  - evolution of the sedimentary bed : *deposition layer managment*
- *morphodynamic* coupling.

The calculations are carried out cell by cell by considering most of the sedimentary variables at the center of the cell. The majority of the calculations are therefore carried out in 1DV. Certain calculations must however be carried out taking into account the adjacent meshes:

- calculate skin stress has current are computed on the mesh edges
- calculate the slope of the bottom and the coefficients necessary to take into account its effect on transport by bedload
- calculate the correction of horizontal sand fluxes
- calculate the fluxes entering a cell induced by bedload and suspension transport

TODO : add a scheme to explain the two main phases : erosion/deposit

Roughness length

Mustang use **grain roughness length** to evaluate moving conditions of particles.

Mustang can also compute a **form roughness length** to account of ripple effect on flow and transmit it to the hydrodynamic code (here CROCO)

The **grain roughness length** could be :

- constant in time and uniform in space with **l_z0seduni** = .TRUE., in this case, the skin roughness length is equal to z0seduni namelist value (see *namelist namsedim_bottomstress*)
- variable in time and space with **l_z0seduni** = .FALSE., in this case, the skin roughness length is computed at each time step from sediment bed composition in each cell (i,j) :
  - if bathymetry is not defined in the cell : z0sed = z0sedmud (see *z0sedmud in namsedim_bottomstress*) (to avoid division by zero in skinstress evaluation when neighbour cells are used)
  - if bathymetry is defined in the cell :
    - if sediment is not present, then z0sed = z0sedbedrock (see *z0sedbedrock in namsedim_bottomstress*)
    - if sediment is present,
      - if there is only mud in the superficial layer z0sed = z0seduni
      - if there is sand or gravel, Nikuradse formulation is used (z0sed = diam/12) with diam corresponding to the ponderate sum of gravel and sand diameter

        $diam = \sum_{n=igrav1}^{isand2} cv_{sed}(n,kmax)/c_{sedtot}(n,kmax) * diameter(n)$

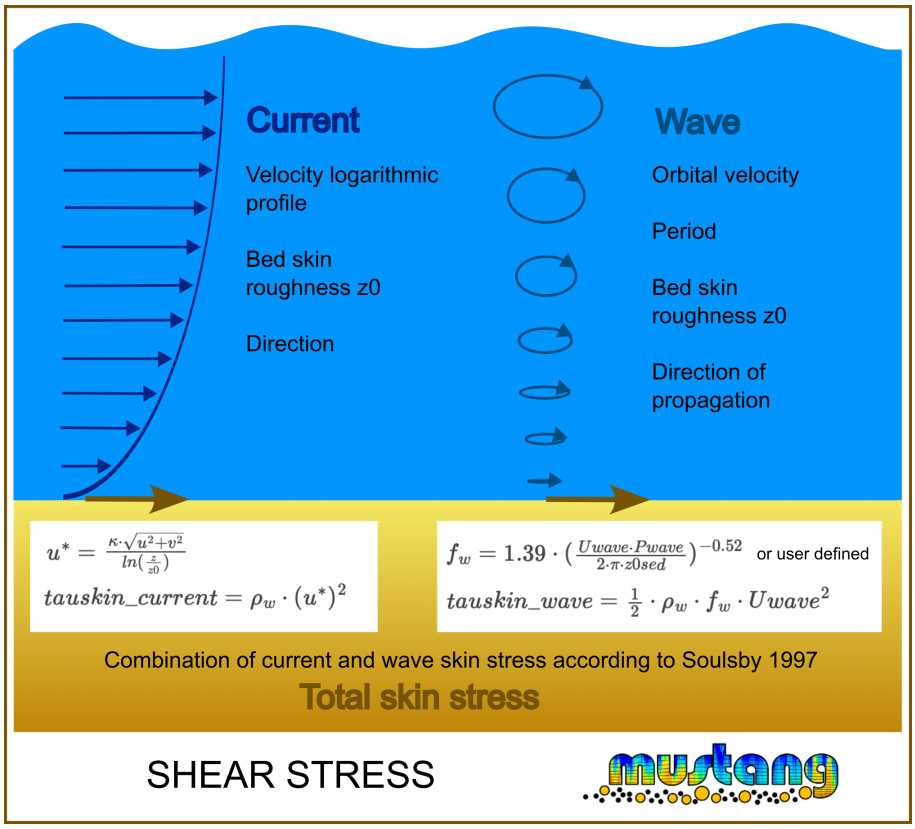
The **form roughness length** is computed and sent to CROCO only if l_z0hydro_coupl (at each time step) or l_z0hydro_coupl_init (just at initialization) :

- if there is no sediment, z0hydro = z0_hydro_bed (see *z0_hydro_bed in namsedim_bottomstress*)
- if there is sediment :
  - if there is more than 30% of mud in the first centimeter of sediment, z0hydro = z0_hydro_mud (see *z0_hydro_mud in namsedim_bottomstress*)
  - else the ponderate diameter of sand is used and z0hydro = coef_z0_coupl * ponderate diameter of sand (see *coef_z0_coupl in namsedim_bottomstress*). Note that gravel are not taking into account here

Shear stress

The shear stress skin friction is computed following steps :

- compute current skin friction
- if cpp keys WAVE_OFFLINE is activated, compute wave skin friction
- if cpp keys WAVE_OFFLINE is activated, compute combination between current and wave skin friction



Current skin friction

Current skin friction is computed from the friction velocity using a logarithmic profile. The friction velocity u^* is computed from roughness length z0 and current component (bottom (u,v) or barotropic (ubar,vbar) using a logarithmic profile :

$u^* = \frac{\kappa \cdot \sqrt{u^2+v^2}}{ln(\frac{z}{z0})}$ with z, height of the bottom cell.

or

$u^* = \frac{\kappa \cdot \sqrt{ubar^2+vbar^2}}{ln(\frac{z}{e \cdot z0})}$ with z, the water height.

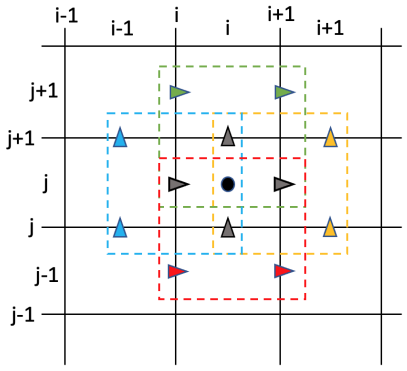
Current skin friction is compute using : $tauskin\_current = \rho_w \cdot (u^*)^2$

The following option are available via cpp keys :

- **default** : compute u^* components at (u,v) locations first and then at the center of the cell. This option use 12 points of current components (see figure below).
- **key_tauskin_c_center** : compute u^* directly at (rho) location (center of the cell) using immediate u,v components. This option use 4 points of current components (see figure below).
- **key_tauskin_c_ubar** : compute u^* using ubar,vbar value instead of bottom layer u,v values.
- **key_tauskin_c_upwind** : depending on current direction, use only component upwind (not combinable with **key_tauskin_c_center**). This option use 8 to 12 points of current components (see figure below).

- **BBL** : computation is done via BBL module of CROCO using constant roughness (from 160 microns diameter). **This option is not recommended** with MUSTANG due to constant roughness.

default : 12 u,v points to compute in cell i,j



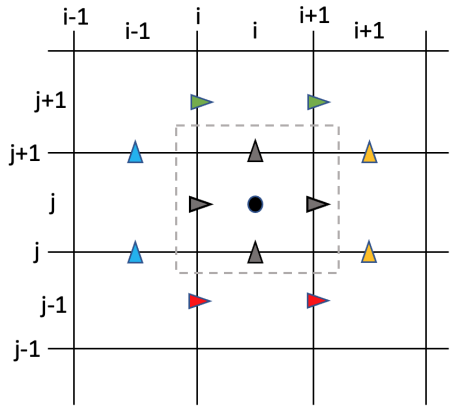
$$ustar2\_u(i,j) = \left[\frac{\kappa}{\log(z/z_0)}\right]^2 \sqrt{u(i,j)^2 + \left[\frac{v(i,j)+v(i-1,j)+v(i,j+1)+v(i-1,j+1)}{4}\right]^2}\, u(i,j)$$

$$ustar2\_v(i,j) = \left[\frac{\kappa}{\log(z/z_0)}\right]^2 \sqrt{v(i,j)^2 + \left[\frac{u(i,j)+u(i+1,j)+u(i,j-1)+u(i+1,j-1)}{4}\right]^2}\, v(i,j)$$

$$tauskin\_x(i,j) = \rho\ (ustar2\_u(i,j) + ustar2\_u(i+1,j)) / 2$$
$$tauskin\_y(i,j) = \rho\ (ustar2\_v(i,j) + ustar2\_v(i,j+1)) / 2$$

$$tauskin\_c(i,j) = \sqrt{(tauskin\_x(i,j))^2 + (tauskin\_x(i,j))^2}$$

key_tauskin_c_center : 4 u,v points to compute in cell i,j



$$Urho(i,j) = \left[\frac{u(i,j)+u(i+1,j)}{2}\right]$$

$$Vrho(i,j) = \left[\frac{v(i,j)+v(i,j+1)}{2}\right]$$

$$speed(i,j) = \sqrt{Urho(i,j)^2 + Vrho(i,j)^2}$$

$$tauskin\_c(i,j) = \rho \left[\frac{\kappa}{\log(z/z_0)}\right]^2 speed^2$$

Wave skin friction

Wave skin friction is computed using Soulsby (1997) formula.

$tauskin\_wave = \frac{1}{2} \cdot \rho_w \cdot f_w \cdot Uwave^2$

With

- f_w equal to **fricwav** *namelist namsedim_bottomstress* value or computed from *WAVE_OFFLINE file* variables ($Uwave$ and $Pwave$) and roughness if **l_fricwave** *namelist namsedim_bottomstress* value is True using $f_w = 1.39 \cdot (\frac{Uwave \cdot Pwave}{2 \cdot \pi \cdot z0sed})^{-0.52}$
- $Uwave$ and $Pwave$ reading from *WAVE_OFFLINE file*

Combinaison of current and wave stresses

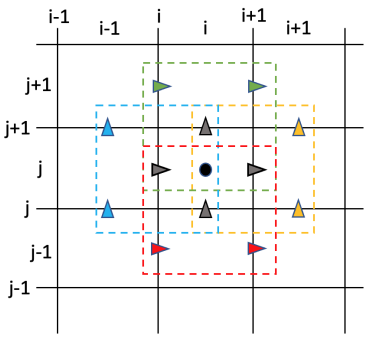
Combinaison of current and wave stresses is done from Soulsby (1997) (Dynamics of marine sands - eq.69 and 70 page 92), $tauskin$ equal to tau_max

$$tau\_mean = tauskin\_current \cdot \left(1 + 1.2 \cdot \left(\frac{tauskin\_wave}{tauskin\_current + tauskin\_wave}\right)^{3.2}\right)$$

$$tau\_max = \sqrt{(tau\_mean + tauskin\_wave \cdot |cos(phi)|)^2 + (tauskin\_wave \cdot |sin(phi)|)^2}$$

With phi angle between current and wave.

key_tauskin_c_upwind : 8 to 12 u,v points to compute in cell i,j depending on current direction



$$ustar2\_u(i,j) = \left[\frac{\kappa}{\log(z/z_0)}\right]^2 \sqrt{u(i,j)^2 + \left[\frac{v(i,j)+v(i-1,j)+v(i,j+1)+v(i-1,j+1)}{4}\right]^2}\, u(i,j)$$

$$ustar2\_v(i,j) = \left[\frac{\kappa}{\log(z/z_0)}\right]^2 \sqrt{v(i,j)^2 + \left[\frac{u(i,j)+u(i+1,j)+u(i,j-1)+u(i+1,j-1)}{4}\right]^2}\, v(i,j)$$

If u(i,j) >0 & u(i+1,j)>0 : tauskin_x(i,j) = ρ ustar2_u(i, j)
else if u(i,j) <0 & u(i+1,j)<0 : tauskin_x(i,j) = ρ ustar2_u(i+1, j)
else : tauskin_x(i,j) = ρ (ustar2_u(i, j) + ustar2_u(i+1, j)) / 2

If v(i,j) >0 & v(i,j+1)>0 : tauskin_y(i,j) = ρ ustar2_v(i, j)
else if v(i,j) <0 & v(i,j+1)<0 : tauskin_y(i,j) = ρ ustar2_v(i, j+1)
else : tauskin_y(i,j) =ρ (ustar2_v(i, j) + ustar2_v(i, j+1)) / 2

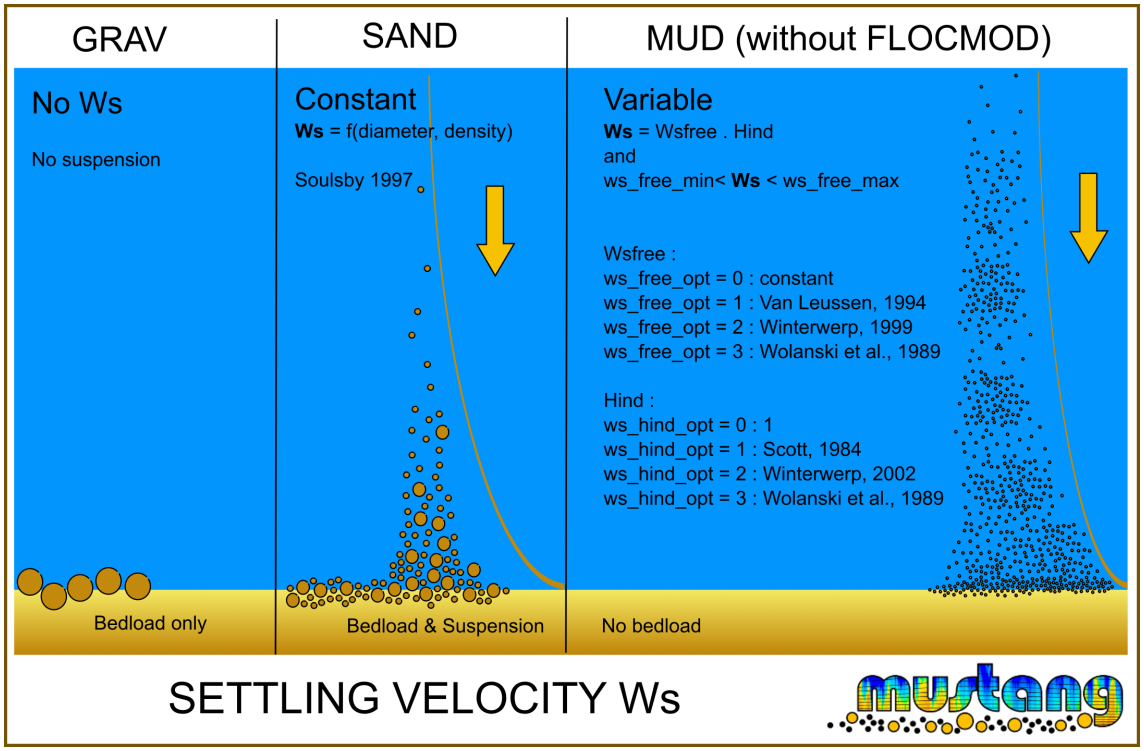
$$tauskin\_c(i,j) = \sqrt{(tauskin\_x(i,j)^2 + (tauskin\_x(i,j))^2}$$

Note: abs() is used on cos() and sin() because of alternative direction of tau_w in the vector addition of tau_c and tau_w (see Sousby 1997 (Dynamics of marine sands - figure 16 page 89))

Settling

The settling process is taken into account during the advection-diffusion scheme in the water column and in the exchange from water to sediment via the deposit fluxes.

Settling velocity modelling strategy



The settling velocity is the main variable of the settling process and is moddelled differently for each substance type :

- GRAV : gravels are not transported in suspension, they have no settling velocity because they are not in the water column

- SAND : sands have a constant settling velocity directly compute from Soulsby (1997) using their diameters defined in *SUBSTANCE namelist &nmlsands* (diam_n).

  $Ws = 10^{-6} \cdot \frac{(107.33+1.049 \cdot Dstar^3)^{0.5}-10.36}{D}$

  With $Dstar = D \cdot 10^4 \cdot (g \cdot (\frac{\rho_s}{\rho_w} - 1))^{\frac{1}{3}}$ the dimensionless diameter of sediment and *D* diameter of sediment class, ρ_w water density and ρ_s sediment density.

  The resulting settling velocity could be high.

- MUD and Non Constitutive Particulate subtances : the settling velocity can vary in time and space depending on the parameters chosen by user in *SUBSTANCE namelist &nmlmuds*: ws_free_opt_n(), ws_free_min_n(), ws_free_max_n(), ws_free_para_n(1:4,num substance), ws_hind_opt_n(), ws_hind_para_n(1:2,num substance)) or by cpp key for flocculation module (see *Flocculation*)

  If flocculation module is not used, settling velocity is the result of the choice on ws_free_opt_n and ws_hind_opt_n values in *SUBSTANCE namelist &nmlmuds* and is computed for every cell "i,j" and layer "k" in the water domain.

  $Ws = max(ws\_free\_min\_n\ ;\ min(ws\_free\_max\_n\ ;\ Ws_{free} \cdot Hind))$

  See appropriate chapters for details on *free settling velocity* and *hindered settling parameter*.

- Sorbed substances : the settling velocity of sorbed substance is the same as the particulate susbtance to which it is associated

- Dissolved and fixed substances : no settling velocity

Free settling velocity

Ws_{free} the free settling velocity is computed from :

- if ws_free_opt_n = 0 : constant value, $Ws_{free} = ws\_free\_min\_n$
- if ws_free_opt_n = 1 : formulation of Van Leussen, 1994, $Ws_{free} = kC^m \cdot \frac{1+aG}{1+bG^2}$

  With :

  - $k = ws\_free\_para\_n(1)$ (= 0.0005 in the reference)
  - $m = ws\_free\_para\_n(2)$ (= 1.2 in the reference)
  - $a = ws\_free\_para\_n(3)$ (= 0.3 in the reference)
  - $b = ws\_free\_para\_n(4)$ (= 0.09 in the reference)
  - C = Sum of concentration of MUD substances in the layer
  - $G = \sqrt{\frac{turbulence\ dissipation\ rate}{vertical\ viscosity\ coefficient}}$ = turbulence energy

- if ws_free_opt_n = 2 : formulation of Winterwerp, 1999, $Ws_{free} = \frac{1}{18} \cdot \frac{(\rho_s-\rho_w)g}{\rho_w \nu} \cdot Dp^{3-nf} \cdot De^{nf-1}$

  With :

  - $De = max(Dp + \frac{ka \cdot C}{kb \cdot \sqrt{G}}\ ;\ \sqrt{\frac{\nu}{G}})$
  - $Dp = ws\_free\_para\_n(1)$ = Primary Particle Diameter, (= 4.10-6 in the reference)
  - $ka = ws\_free\_para\_n(2)$ = aggregation factor, (= 14.6 in the reference)
  - $kb = ws\_free\_para\_n(3)$ = breakup factor, (= 30000 in the reference)
  - $nf = ws\_free\_para\_n(4)$ = fractal dimension, (= 2 in the reference)
  - C = Sum of concentration of MUD substances in the layer
  - $G = \sqrt{\frac{turbulence\ dissipation\ rate}{vertical\ viscosity\ coefficient}}$ = turbulence energy
  - $\nu = 0.00000102\ m^2/s$ = water kinematic viscosity
  - g = gravity

  - ρ_s = sediment density
  - ρ_w = water density
- if ws_free_opt_n = 3 : formulation of Wolanski et al., 1989, $Ws_{free} = kC^m$

  With :
  - $k = ws\_free\_para\_n(1)$ (= 0.01 in the reference)
  - $m = ws\_free\_para\_n(2)$ (= 2.1 in the reference)
  - C = Sum of concentration of MUD substances in the layer

Hindered settling parameter

$Hind$ the hindered settling parameter is computed from :

- if ws_hind_opt_n = 0 : no hindered effect, $Hind = 1$
- if ws_hind_opt_n = 1 : formulation of Scott, 1984, $Hind = (1 - \phi)^m$

  With :
  - $\phi = min(1\ ;\ \frac{C}{cgel})$
  - C = Sum of concentration of MUD substances in the layer
  - $cgel = ws\_hind\_para\_n(1)$ (= 40 in the reference)
  - $m = ws\_hind\_para\_n(2)$ (= 4.5 in the reference)
- if ws_hind_opt_n = 2 : formulation of Winterwerp, 2002, $Hind = (1 - \phi_v)^m \cdot \frac{(1-\phi)}{(1+2.5\phi_v)}$

  With :
  - $\phi = \frac{C}{\rho_s}$
  - If ws_free_opt_n is not 2 then $\phi_v = \frac{C}{cgel}$
  - If ws_free_opt_n is 2 then $\phi_v = \phi \cdot (\frac{De}{Dp})^{3-nf}$
    - $De = max(Dp + \frac{ka \cdot C}{kb \cdot \sqrt{G}}\ ;\ \sqrt{\frac{v}{G}})$
    - $Dp = ws\_free\_para\_n(1)$ = Primary Particle Diameter, (= 4.10-6 in the reference)
    - $nf = ws\_free\_para\_n(4)$ = fractal dimension, (= 2 in the reference)
  - $cgel = ws\_hind\_para\_n(1)$ (= 40 in the reference)
  - $m = ws\_hind\_para\_n(2)$ (= 1 in the reference)
  - C = Sum of concentration of MUD substances in the layer
  - ρ_s = sediment density
- if ws_hind_opt_n = 3 : formulation of Wolanski et al., 1989, this formulation has to be combined with ws_free_opt_n = 3.

  If ws_free_opt_n is not 3 then no hindered effect $Hind = 1$

  If ws_free_opt_n is 3 then $Hind = \frac{1}{(C^2 + b_w^2)^{m_w}}$

  With :
  - $b_w = ws\_hind\_para\_n(1)$ (= 2 in the reference)
  - $m_w = ws\_hind\_para\_n(2)$ (= 1.46 in the reference)
  - C = Sum of concentration of MUD substances in the layer

Treatment of high settling velocities : SAND variables

For high settling velocities, the major part of the sediment are concentrated in a thin layer near bottom. The thickness of the bottom layer in water could be unadapted to represent correctly the concentration at bottom. This could lead to an underestimation of deposit fluxes and horizontal transport fluxes.

Furthermore, to avoid numerical instabilities, vertical transport is computed using sub time steps. The number of sub time steps depends on the settling velocity of each substance. For high settling velocities, this could be time consuming.

In MUSTANG, these issues are considered for SAND variables only. It is considered that MUD have too low settling velocities and GRAVEL are not transported in suspension.

That is why three features have been developped for SAND variables in MUSTANG to treat these issues :

- vertical deposit fluxes are corrected considering a Rouse profile in the bottom layer in water. The concentration is extrapolated at a given reference height (parameter **aref_sand** of *MUSTANG namelist*) to obtain a correct deposit flow.
- horizontal advection fluxes could, as an option, be corrected considering a Rouse profile of concentration and the current logarithmic gradient near bottom. This option is activated by the cpp key **MUSTANG_CORFLUX**
- SAND sediments could, as an option, be considered as 2D variables with the cpp key **key_sand2D**. When this ccp key **key_sand2D** is activated, a boolean **l_sand2D** in *SUBSTANCE namelist &nmlsands* specify for each SAND substance if it has to be treated as a 2D variable and not 3D. This option makes it possible to treat the fall of SAND-type sediments by considering that the transport in suspension only takes place in the bottom layer (2D sand transport in a single layer). This makes it possible to skip the vertical transport and mixing for this susbtance and save calculation time. As an option (**l_outsandrouse** in *SUBSTANCE namelist &nmlsands*), the Rouse profile is reconstructed in the outputs.

The Rouse profile used in those three features involves the ratio Ws/u* to calculate the Rouse number (Z in the formulation below). The formulations used are those proposed by Julie Vareilles (activity report of the post-doctorate Consequences of Climate Change on Ecogeomorphology of Estuaries, March 2013)

Rouse profile:

$C(z) = C(a) \cdot (\frac{h-z}{z} \cdot \frac{a}{z-a})^Z$

With :

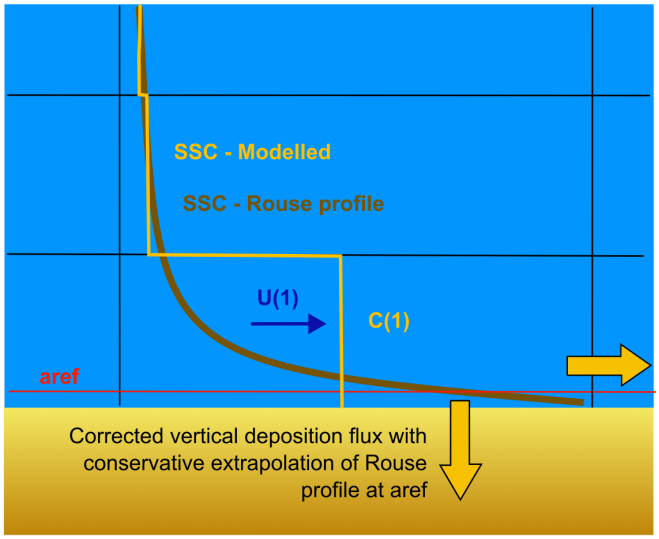
- a : reference height
- h : water height
- Z : Rouse number

For $\frac{W_s}{u_*} < 0.1 : \beta = 1, Z = \frac{W_s}{\kappa \cdot u_*}$

For $0.1 <= \frac{W_s}{u_*} < 0.75 : \beta = 1 + 2(\frac{W_s}{u_*})^2, Z = \frac{W_s}{\beta \cdot \kappa \cdot u_*}$

For $0.75 <= \frac{W_s}{u_*} < 1.34 : Z = 0.35\frac{W_s}{u_*} + 0.727$

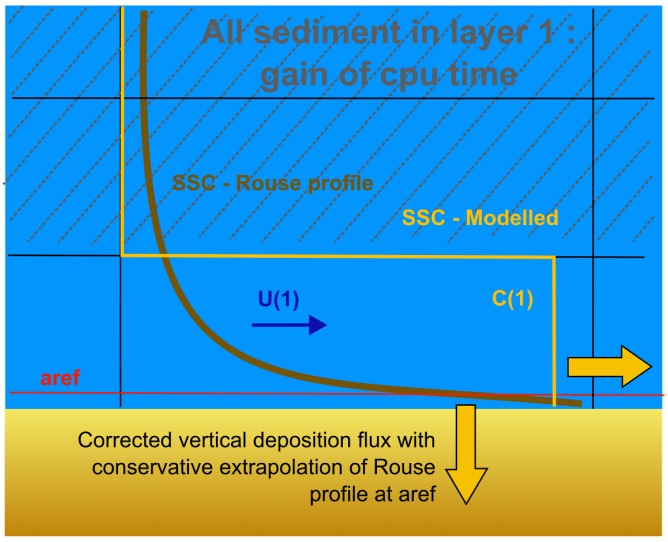
For $1.34 <= \frac{W_s}{u_*} : Z = 1.2$



#MUSTANG_CORFLUX
Horizontal advection fluxes correction factor considering a Rouse profile of concentration and a logarithmic profile of current :

$$\text{corflux} = \frac{\int_{aref}^{zw(1)} C_{rouse}\ u_{log}\ dz}{u(1)C(1)\Delta z(1)}$$

SAND suspension corrections in 3D



#MUSTANG_CORFLUX
Horizontal advection fluxes correction factor considering a Rouse profile of concentration and a logarithmic profile of current :

$$\text{corflux} = \frac{\int_{aref}^{zw(1)} C_{rouse}\ u_{log}\ dz}{u(1)C(1)\Delta z(1)}$$

SAND suspension corrections with **#key_sand2D**

Flocculation with FLOCMOD

Introduction

Suspended particulate matter (SPM) dynamics in estuaries and coastal seas is driven by flocculation processes. These processes modify floc sizes and floc densities, and hence their settling velocity. This parameter is crucial when modelling SPM transport in coastal systems. It can be set as a constant value, or evaluated though an empirical function (such as Van Leussen relationship, see *settling velocity chapter*) to mimic flocculation dynamics with processes "at equilibrium".

FLOCMOD is a 0D size-class-based module developed by Ifremer to simulate the explicitly flocculation processes (See Verney et al., 2011). It based on the population equation system originally proposed by Smoluchowski (1917). As a size class-based model, this means that the floc size distribution is represented by a discrete number of sediment classes of increasing sizes. This module simulates the effect of turbulence and SPM concentration and flocculation processes, and uses the fractal approach to represent main floc properties (floc density, floc mass, floc settling velocity).

Future developments are scheduled to include seasonal variability in organic matter content and hence modulate flocculation processes, and floc resistance to breakup for instance.

Module description

FLOCMOD in CROCO

FLOCMOD is available in CROCO using the SEDIMENT (USGS) module or the MUSTANG module. Using MUSTANG, **FLOCMOD** is activated with the cppkey **#key_mustang_flocmod**. The MUSTANG specific cppkeys must also be activated.

Floc size classes are defined as mud types in *SUBSTANCE namelist parasubstance_MUSTANG.txt*. The floc diameter in the namelist correspond to the floc size of the given class. Update the number of sediment classes in param.h (ntrc_sub) accordingly. You may also update obc and input files and sediment initial distribution file if you want to prescribe a specific floc size distribution from rivers or open boundary conditions. Floc sizes are discrete classes, which means that floc created by aggregation or fragmentation is redistributed along the two nearest classes (in floc mass) using a mass-conservative interpolation scheme based on inverse distance.

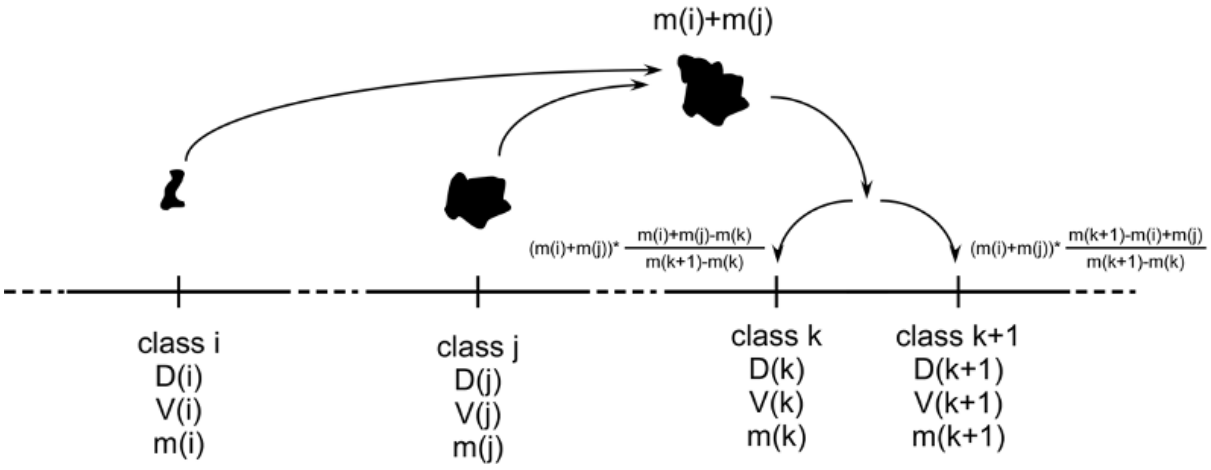


Fig. 1: Flocmod redistribution on discrete classes

FLOCMOD parameters are all defined in *MUSTANG namelist* within the flocmod namelist *namflocmod*. The different parameters are listed and detailed below when describing flocculation processes.

To save computation time, probability kernels are mainly pre-processed before the computation loop, except the fragmentation by collision kernel. Be careful, if **l_COLLFRAG** is activated, computation time can be very significantly impacted.

FLOCMOD processes

Flocculation processes are actually competition between aggregation and breakup mechanisms, driven by turbulence, SPM concentration, and potentially salinity and organic matter content. The last two control parameter

are not yet included in FLOCMOD explicitly. Aggregation is controlled by turbulent shear and differential settling, while fragmentation is exclusively controlled by shear, with different (concurrently usable) options: shear fragmentation, shear erosion, collision-induced fragmentation. All floc interactions are limited to "two-body" interactions. The generic equation given below define all processes involved in FLOCMOD, both in term of gain (G) or loss (L) per class. The main model variable is the number concentration of flocs Nk in a given class k. ASH, ADS and fragmentation by collision can be activated using boolean l_ASH, l_ADS and l_COLLFRAG respectively from the namelist.

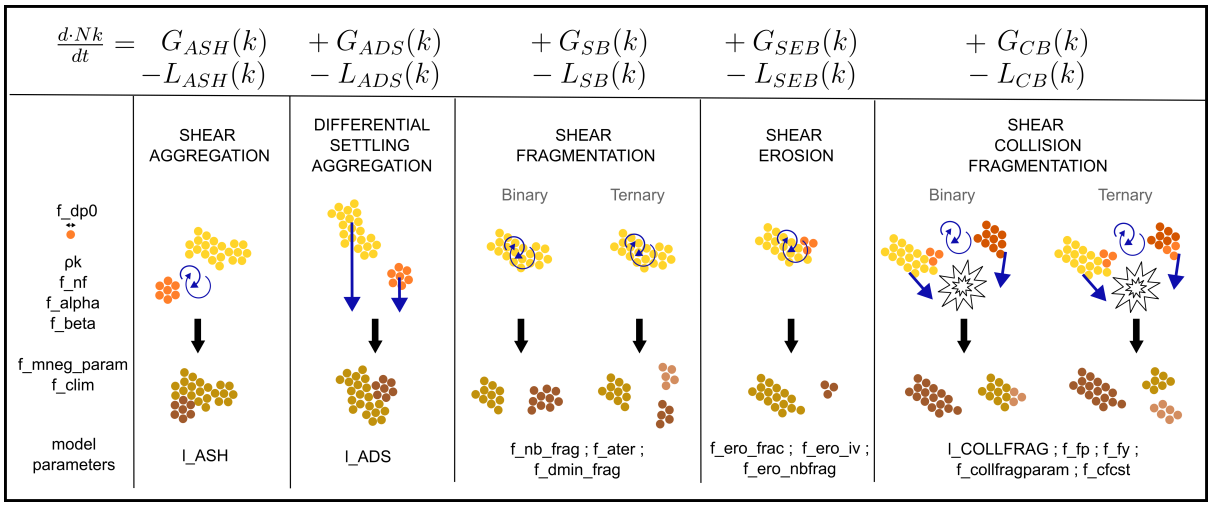


Fig. 2: Flocmod processes

Floc fractal approach

Flocs observed in nature are characterized by a wide range of size and densities, based on mineral intrinsic features, the organic matter content in SPM, and based on hydrological and hydrodynamics factors. In general, small flocs have high excess densities (O(100-1000 kg/m3)), and large flocs low densities (O(10-100kg/m3)). In FLOCMOD, we use the fractal approach to represent floc characteristics (Kranenburg, 1994) : floc size D, floc mass M, floc excess density $\Delta\rho$ ($\rho - \rho_w$). Flocs are formed from N primary particles with a unique size D_p (f_dp0 in the namelist) and density ρ_p (ros from the *SUBSTANCE namelist parasubstance_MUSTANG.txt*). The floc structure (~compacity) is controlled by the fractal dimension n_f (f_nf) such as:

$$N = (\frac{D}{D_p})^n_f$$

$$M = \rho_p \cdot \frac{\pi}{6} \cdot {D_p}^3 \cdot N$$

$$\Delta\rho = (\rho_p - \rho_w) \cdot (\frac{D}{D_p})^{3-n_f}$$

Shear rate G

The shear rate G is an hydrodynamic parameter. If GLS_MIXING cppkey is used in CROCO, G can be directly calculated from ϵ such as :

$$G = \sqrt{\frac{\epsilon}{\nu}}$$

Otherwise, an ϵ vertical profile is calculated from Nezu and Nakagawa and the bottom friction velocity u^* (from MUSTANG) such as :

$$\epsilon = \frac{{u^*}^3}{\kappa h} \frac{h-z}{z}$$

Where h is the water depth and z the distance from bottom.

Shear aggregation (G_{ASH} ; L_{ASH})

These terms correspond to the gain or loss of class k particles when i) two particles in movement (by shear) collide and ii) the collision is efficient, i.e. the newly formed bound between the two particles can withstand the shear induced by the collision. The two-body collision probability function $A_{SH}(i,j)$ is a function of the shear rate G and particle diameters D_i and D_j (McAnally andMehta,2001,2002)

$$G_{ASH}(k) = \frac{1}{2} \sum_{M_i+M_j=M_k} \alpha_{ij} A_{SH}(i,j) n_i n_j$$

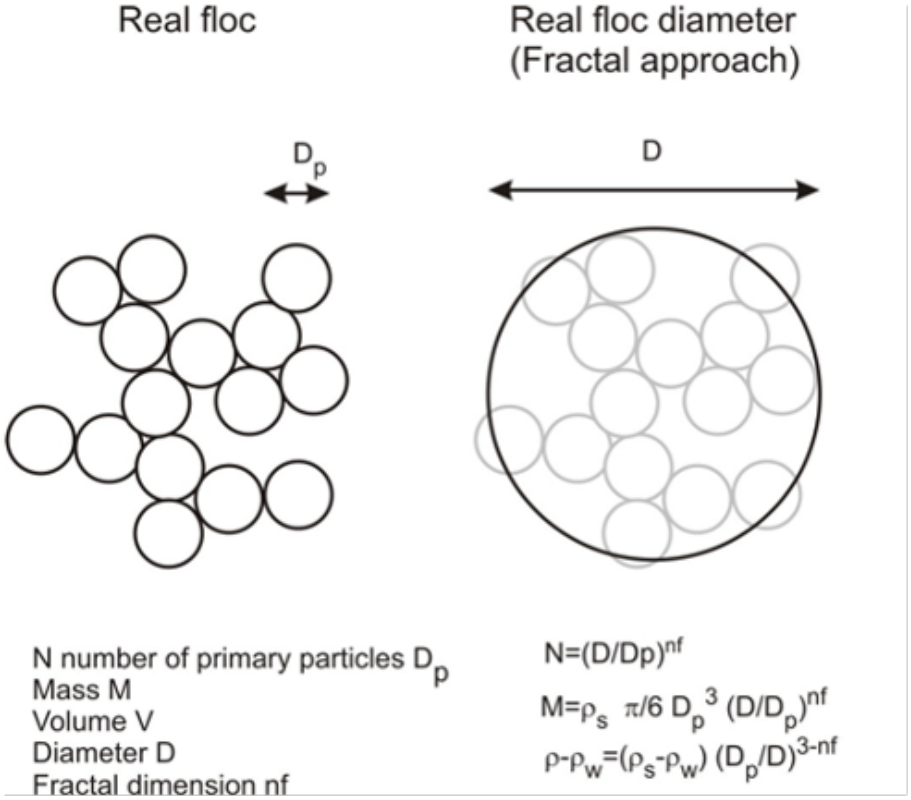


Fig. 3: Flocmod fractal approach

$L_{ASH}(k) = \sum_{i=1}^{nc} \alpha_{ik} A_{SH}(i,k) n_i n_k$

$A_{SH}(i,j) = \frac{G}{6}(D_i + D_j)^3$

α_{ij} is the collision efficiency representing particle cohesiveness, i.e. the physico-chemical forces and the sticking properties of organic matter. In the actual version of FLOCMOD, $\alpha_{ij} = \alpha$ is a constant parameter (between 0 and 1) set in the FLOCMOD namelist.

Differential settling aggregation (G_{ADS} ; L_{ADS})

A_{DS} represents collisions and aggregation that can occur when 2 flocs with different settling velocities can interact during settling. Kernels are similar to G_{ASH} and L_{ASH}, except that the collision probability A_{SH} is substituted with A_{DS} such as :

$A_{DS}(i,j) = \frac{\pi}{4}(D_i + D_j)^2 \mid W_{s,i} - W_{s,j} \mid$

Where $W_{s,i}$ is the floc settling velocity of class i, calculated from Stokes :

$W_{s,i} = \frac{g}{18\mu} \Delta\rho_i {D_i}^2$

Shear fragmentation (G_{SB} ; L_{SB})

Shear fragmentation represents the action of shear (turbulent) forces on flocs, leading to breakup.

$G_{SB}(k) = \sum_{i=k+1}^{nc} FDSB_{ki} B_i n_i$

$L_{SB}(k) = B_k n_k$

$B_i = \beta_i G^{\beta_2} D_i (\frac{D_i - D_p}{D_p})^{\beta_3}$

β_i is the fragmentation rate, indirectly related to yield strength and floc resistance to breakup. In the actual version of FLOCMOD, β_i is a constant set in the FLOCMOD namelist. Following Winterwerp (2002), *beta_2 = 3/2* and *beta_3 = 3 - n_f*.

FDSB is the size distribution function of fragmented flocs by shear. Two modes are available: binary or ternary breakup.

- Binary distribution : fragmentation of a floc with mass m_i in two identical flocs of mass $m_i/2$.

$$FDSB_{ij} = \begin{cases} 2 & \text{if } m_j = \frac{m_i}{2} \\ 0 & \text{otherwise} \end{cases}$$

- Ternary distribution : fragmentation of a floc with mass m_i in one floc of mass $m_i/2$ and 2 flocs of mass $m_i/4$.

$$FDSB_{ij} = \begin{cases} 1 & \text{if } m_j = \frac{m_i}{2} \\ 2 & \text{if } m_j = \frac{m_i}{4} \\ 0 & \text{otherwise} \end{cases}$$

In FLOCMOD, binary fragmentation is activated if f_nb_frag = 2 and f_ater = 0, and ternary fragmentation if f_nb_frag = 2 and f_ater = 0.5; Shear breakup can be only applied from a given floc size to limit fragmentation for small flocs. This can be set in FLOCMOD by changing f_dmin_frag.

Erosion fragmentation

Shear fragmentation by floc erosion is an additional option to represent floc breakup. In this case, flocs are not broken by 2 or 4 but a small fraction of the floc mass is eroded from the parent floc. This mode transfers a part of the shear fragmentation probability to floc erosion using f_ero_frac. This means that (1-f_ero_frac) contributes to binary/ternary fragmentation and f_ero_frac to fragmentation by erosion.

G_{SEB} and L_{SEB} are calculated identically to G_{SB} and L_{SB}, except that the fragmentation distribution changes according to the floc erosion parameter (FDSEB instead of FDSB) :

$$FDSEB_{ij} = \begin{cases} 1 & \text{if } m_j = m_i - f\_ero\_nbfrag \cdot m_i \\ 2 & \text{if } m_j = m_i \\ 0 & \text{otherwise} \end{cases}$$

Collision fragmentation

Collision can not only contribute to form bigger flocs, but instead if turbulence is strong, collision can overpass floc strength and then contribute to break particles by mechanical failure. Hence, when activating this option, part of the collision probability ($A_{SH}(i,j)$) can induce fragmentation.

$G_{CB}(k) = \sum_{i=1}^{nc} \sum_{j=i}^{nc} FDCB_{ij} \cdot A_{SH}(i,j) \cdot n_i \cdot n_j$

$L_{CB}(k) = \sum_{i=1}^{nc} FDCB_{ik} \cdot A_{SH}(i,k) \cdot n_i \cdot n_k$

$FDCB_{ij}$ is the distribution of flocs after collision, and is function of the floc strength $\tau_{y,i}$ and the collision-induced shear stress between floc i and j : $\tau_{collij,i}$.

$\tau_{collij,i} = \frac{8}{\pi} \frac{(G\frac{D_i+D_j}{2})^2}{F_p \cdot {D_i}^2 (D_i+D_j)} \frac{M_i \cdot M_j}{M_i+M_j}$

$\tau_{y,i} = F_y \frac{\Delta\rho_i}{\rho_w}^{\frac{2}{3-nf}}$

Fp and Fy can be user-defined in the FLOCMOD namelist as f_fp and f_fy respectively.

For two-body interactions (i and j), two types of failures are likely to happen and control the expression of $FDBC_{ij}$ (McAnally, 1999):

- Collision fragmentation #1 : $\tau_{y,i} > \tau_{collij,i}$ and $\tau_{collij,i} > \tau_{y,j}$: the collision-induced shear stress exceeds the shear strength of the weakest aggregate only, consequently during the collision, the j floc breaks into two fragments ($F_{j,1}$ and $F_{j,2}$) such as $MF_{j,1} = (1 - cfcst) \cdot Mj$ and $MF_{j,2} = cfcst \cdot Mj$. $F_{j,1}$ is a free fragment while $F_{j,2}$ is bound with the i floc. cfcst is the inter-penetration depth, fixed as 3/16 based on McAnally (1999). This parameter is found in the namelist as f_cfcst.
- Collision fragmentation #2 : $\tau_{y,i} < \tau_{collij,i}$ and $\tau_{collij,i} > \tau_{y,j}$: the shear stress is larger than shear strength of both i and j flocs. Both flocs break into two fragments ($F_{i,1}$; $F_{i,2}$) and ($F_{j,1}$; $F_{j,2}$) such as $[m_F i, 1; m_F j, 1] = (1-cfcst)[M_j; M_j]$ and $[M_F i, 2; m_F j, 2] = cfcst[M_j; M_j]$. Two particles are formed from the two parent flocs $F_{i,1}$ and $F_{j,1}$. A third flocs is formed from the two fragments ($F_{i,2}$ and $F_{j,2}$) that bound during collision.

The fraction of collision contributing to floc breakup is controlled through the parameter f_collfragparam.

FLOCMOD execution

FLOCMOD can be non-conservative for high shear rates and/or high SPM concentration. To prevent for instabilities, FLOCMOD includes a sub-time step algorithm. After mass exchange due to flocculation, FLOCMOD checks if the new floc size distribution is fully positive or null. If true, execution continues. Otherwise, the time step is divided by two and a new floc size distribution is recalculated. This time-step adaptation is applied as long as floc size distribution contains negative mass. Flocculation is then repeated until reaching a cumulated time step corresponding to the CROCO time step.

It is possible to be slightly permissive and allow a small negative mass concentration (f_mneg_param). In this case, the class characterized by negative mass is set to 0 and the "corresponding added mass" is proportionally removed from the positive classes to be mass conservative. This can help to limit time step adaptation, and hence reduce computation costs.

It is also possible to disconnect FLOCMOD when SPM concentration is very low. In FLOCMOD, this concentration threshold (in g/l) is defined in f_clim and set by default to 0.001 g/l.

Erosion process

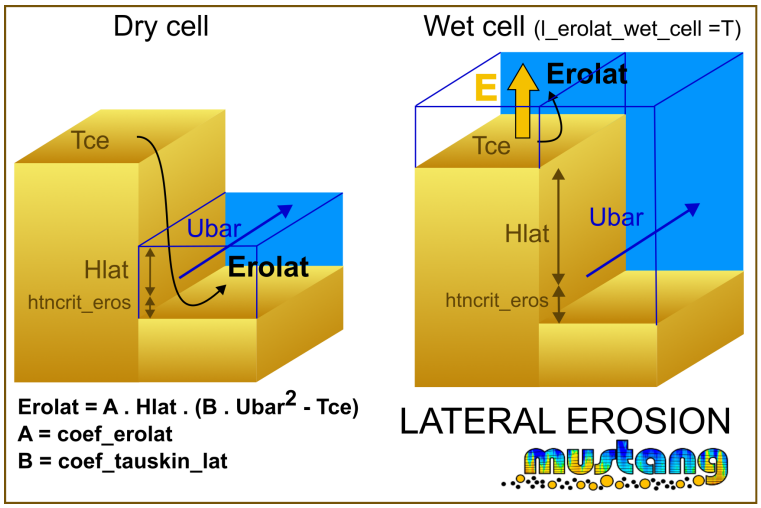
Note: Patience, work in progress, meanwhile see : https://mars3d.ifremer.fr/docs/doc_MUSTANG/doc.MUSTANG.erosion.html

Erosion fluxes

Lateral erosion

key_MUSTANG_lateralerosion

Note: Patience, work in progress



Erosion, layer managment

Deposit process

Note: Patience, work in progress, meanwhile see : https://mars3d.ifremer.fr/docs/doc_MUSTANG/doc.MUSTANG.deposit.html

Deposit processes are treated implicitly in water transport equations. First, deposition flux trends are evaluated for each variable before advection computations.

Then after advection resolution, effective deposit is computed with new concentrations in water (estimated after transport and settling) and sediment layers are updated (see *Deposition layer managment*)

Deposit fluxes

TODO : add formulation used for deposit fluxes

Sliding fluxes

A sliding process of the fluid mud is implemented. Only MUD sediments are concerned by this feature. The modelling strategy consists in :

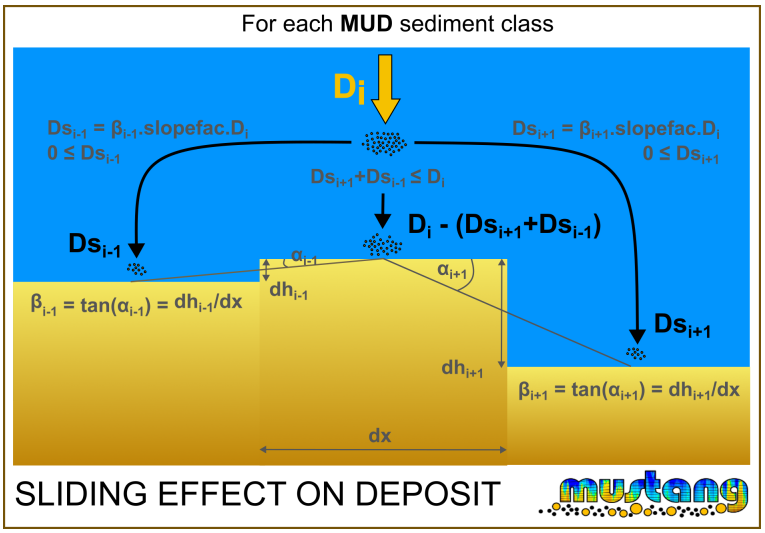
- compute the part of mud which slides if the slope is steep
- deposit this part towards lower neighboring cells according to the slope

To activate this behavior, the cppkey **#key_MUSTANG_slipdeposit** must be define and the **slopefac** value in MUSTANG namelist *&namsedim_deposition* must be greater than 0.

The part of mud which slides on each cell limit is the product of the slope by the deposit fluxe for each mud class in the cell and by the factor slopefac.

No sediment slides if the slope is not positive.

The sum of sliding sediment over the four cell's limits could not be greater than the deposit flux computed in the cell.



Deposit layer managment

Note: Patience, work in progress

Consolidation

Cpp keys involved : #key_MUSTANG_add_consol_outputs

> **Warning:** NOT TESTED YET IN CROCO Documentation to come after. Meanwhile : https://mars3d.ifremer.fr/docs/doc_MUSTANG/doc.MUSTANG.consol.html

Diffusion within sediment and at interface

Cpp keys involved : #key_noTSdiss_insed #key_nofluxwat_IWS

> **Warning:** NOT TESTED YET IN CROCO Documentation to come after. Meanwhile : https://mars3d.ifremer.fr/docs/doc_MUSTANG/doc.MUSTANG.diffu.html

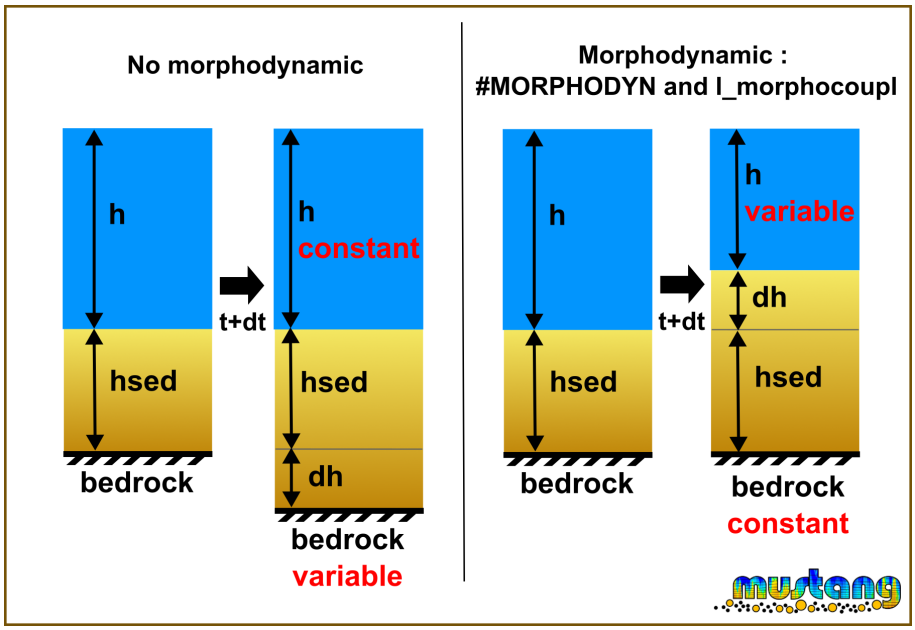
Bioturbation

> **Warning:** NOT TESTED YET IN CROCO Documentation to come after. Meanwhile : https://mars3d.ifremer.fr/docs/doc_MUSTANG/doc.MUSTANG.bioturb.html

Morphodynamic

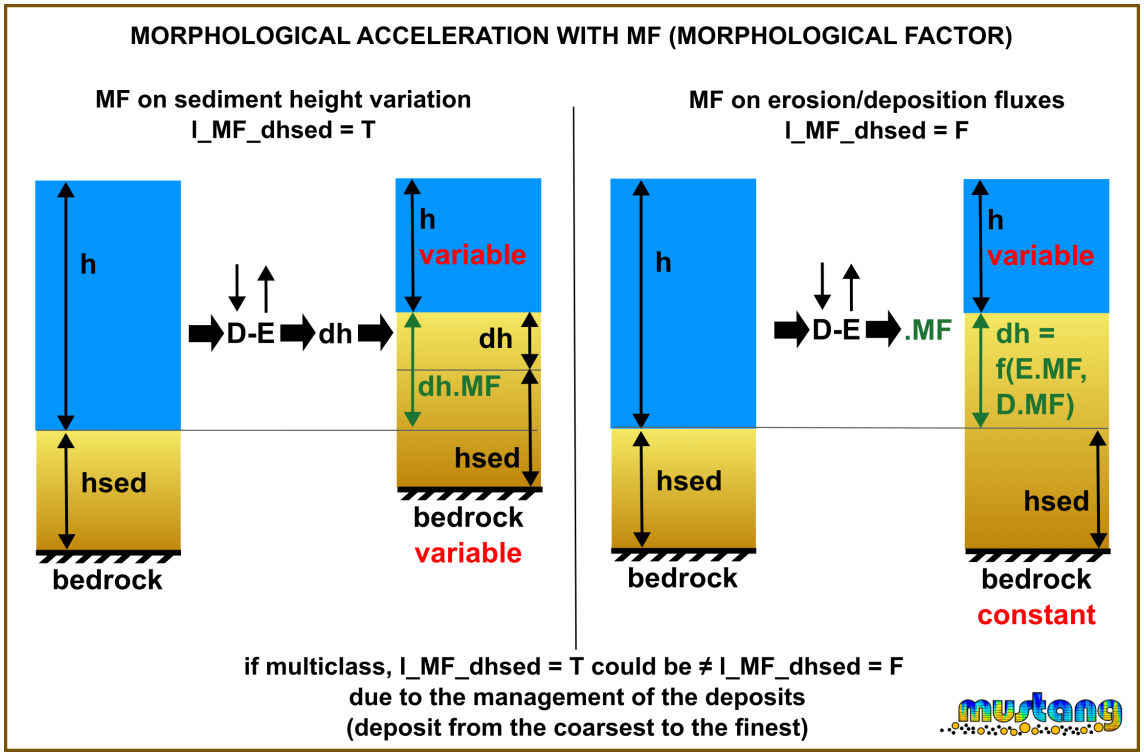
To activate morphodynamic means that the bathymetry used in the hydrodynamic model will evoluate with time. The following figure shows the difference between a non-morphodynamic (ie morphostatic) simulation and a morphodynamic simulation.

The user needs to activate cpp key #MORPHODYN and to set to true the boolean l_morphocoupl (see *&namsedim_morpho*) to run in morphodynamic mode.



The user can also accelerate morphologic evolution by using MF parameter (see *&namsedim_morpho*). In this case, two option are available to accelerate the changes :

- MF could amplified directly sediment height variation (l_MF_dhsed = T) on water height. In this case, sediment height and the bed layers compositions are not modified. Bedrock location is modified.
- MF could amplified erosion/deposition fluxes (l_MF_dhsed = F). In this case, depending on sediment classes in the simulation, the bed layer composition could be different from the case without MF or with (l_MF_dhsed = T) as coarse sediment settled before small one. Sediment height and bed layers compositions are modified but bedrock location is maintained.



FAQ and known issues

Note: Patience, work in progress

- V1 or V2 what difference ?
- V2 with no sed
- River + l_sand2D
- Sand in 3D : CPU time

Coarse sediment in MUSTANG

In MUSTANG V1 (#key_MUSTANG_V2 is not defined), GRAVEL can not go in suspension, they will not move. Another way to deal with GRAVEL in V1 is to declare them as SAND. They will not travel far anyway in suspension, but at least they will impact the sediment dynamics. **If their diameter is greater than 2 mm, they will not impact the mean critical bed shear stress** (i.e. common critical bed shear stress for all sediment classes in V1).

Nevertheless, MUSTANG V2 is recommended to deal with coarse sediment.

Not yet implemented features

The feature related to the subroutine bathy_actu_fromfile work for MARS3D but have not been translate for CROCO yet.

The influence of high concentrations of particles on water density have not been coded yet.

Consolidation, bioturbation, flocculation , diffusion within sediment and at interface are code but have not been tested yet.

MUSTANG references

Mengual Baptiste, Le Hir Pierre, Cayocca Florence, Garlan Thierry (2017). Modelling Fine Sediment Dynamics: Towards a Common Erosion Law for Fine Sand, Mud and Mixtures . Water , 9(8), 564 (1-24) . Publisher's official version : 2017_official Open Access version : 2017_openaccess

Grasso Florent, Le Hir Pierre, Bassoullet Philippe (2015). Numerical modelling of mixed-sediment consolidation . Ocean Dynamics , 65(4), 607-616 . Publisher's official version : 2015_official Open Access version : 2015_openaccess

Le Hir Pierre, Cayocca Florence, Waeles Benoit (2011). Dynamics of sand and mud mixtures: a multiprocess-based modelling strategy. Continental Shelf Research, 31(10), S135-S149. Publisher's official version : 2011_official Open Access version : 2011_openaccess

15.3 Biogeochemical models

CROCO comes with series of biogeochemical (BGC) models of increasing complexity, from relatively simple 5- or 7-component NPZD (Gruber et al., 2006, 2011) and N2P2Z2D2 BioEBUS model (Gutknecht, 2013) that proved well suited to upwelling regions to 24-component PISCES (Aumont et al., 2005).

BioEBUS is a nitrogen-based model (Fig. 1) derived from a N2P2Z2D2 evolution of ROMS NPZD model (Gruber et al., 2006, 2011) and accounting for the main planktonic communities in upwelling ecosystems associated oxygen minimum zones (OMZs). It is validated in Gutknecht et al. (2013) using available satellite and in situ data in the northern part of the Benguela upwelling system. In this model, phytoplankton and zooplankton are split into small (PS and ZS: flagellates and ciliates, respectively) and large (PL and ZL: diatoms and copepods, respectively) organisms. Detritus are also separated into small and large particulate compartments (DS and DL). A semi-labile dissolved organic nitrogen (DON) compartment was added since DON can be an important reservoir of OM and can potentially play an important role in supplying nitrogen or carbon from the coastal region to the open ocean (Huret et al., 2005). The pool of dissolved inorganic nitrogen is split into nitrate (NO3-), nitrite (NO2-) and ammonium (NH4+) species to have a detailed description of the microbial loop: ammonification/nitrification processes under oxic conditions, and denitrification/anammox processes under suboxic conditions (Yakushev et al., 2007). These processes are directly oxygen dependent, so an oxygen (O2) equation was also introduced in BioEBUS with the source term (photosynthesis), sink terms (zooplankton respiration, bacteria re-mineralisation) and sea–air O2 fluxes following Pena et al. (2010) and Yakushev et al. (2007). To complete this nitrogen-based model, nitrous oxide (N2O) was introduced using the parameterization of Suntharalingam et al. (2000, 2012). It allows determining the N2O production under oxygenated conditions and at low-oxygen levels, mimicking the N2O production from nitrification and denitrification processes. The SMS terms of BioEBUS and parameter values are described in detail in Gutknecht et al. (2013).

PISCES was developed for NEMO (the French ocean climate model). It was implemented in CROCO for its supposed suitability for a wide range of oceanic regimes. PISCES currently has five modeled limiting nutrients for phytoplankton growth: Nitrate and Ammonium, Phosphate, Silicate and Iron. Phosphate and nitrate+ammonium are linked by constant Redfield ratios but the nitrogen pool undergoes nitrogen fixation and denitrification. Four living compartments are represented: two phytoplankton size-classes/groups corresponding to nanophytoplankton and diatoms, and two zooplankton size classes which are micro-zooplankton and mesozooplankton. For phytoplankton, prognostic variables are total biomass, the iron, chlorophyll and silicon contents. This means that the Fe/C, Chl/C and Si/C ratios of both phytoplankton groups are fully predicted by the model. For zooplankton, only the total biomass is modeled. For all species, the C/N/P/O2 ratios are supposed constant and are not allowed to

vary. The Redfield ratio O/C/N/P is set to 172/122/16/1. In addition, the Fe/C ratio of both zooplankton groups is kept constant. No silicified zooplankton is assumed. The bacterial pool is not yet explicitly modeled. There are three non-living compartments: semi-labile dissolved organic matter, small and big sinking particles. The iron, silicon and calcite pools of the particles are explicitly modeled and their ratios are allowed to vary. The sinking speed of the particles is not altered by their content in calcite and biogenic silicate ("The ballast effect"). The latter particles are assumed to sink at the same speed as big organic matter particles. All the non-living compartments experience aggregation due to turbulence and differential settling. In addition to the ecosystem model, PISCES also simulates dissolved inorganic carbon, total alkalinity and dissolved oxygen. The latter tracer is also used to define the regions where oxic or anoxic remineralization takes place. see Aumont et al. (2005) in the documentation section for details.

Related CPP options:

PISCES	Activate 24-component PISCES biogeochemical model
BIO_NChlPZD	Activate 5-component NPZD type model
BIO_N2PZD2	Activate 7-component NPZD type model
BIO_BioEBUS	Activate 12-component NPZD type model

Preselected options:

```
# ifdef BIOLOGY
# undef PISCES
# define BIO_NChlPZD
# undef BIO_N2ChlPZD2
# undef BIO_BioEBUS
# endif
```

15.4 Lagrangian floats

CHAPTER

SIXTEEN

COUPLING CROCO WITH OTHER MODELS

CROCO is coupled to atmospheric and wave models through the OASIS-MCT (Ocean-Atmosphere-Sea-Ice-Soil, Model Coupling Toolkit) coupler developed by CERFACS (Toulouse, France). This coupler allows the atmospheric, oceanic, and wave models to run at the same time in **parallel**, it **exchanges** variables, and performs **grid interpolations** and **time transformations** if requested. OASIS is not an executable file, but a set of libraries providing functions which are called in the models themselves. The variables exchanged by the coupler, as well as the grid interpolations are specified through a namelist file (called `namcouple`).

CROCO can therefore be coupled to any code in which OASIS-MCT is implemented. Non-exhaustively, here are some models including OASIS-MCT, that can be coupled to CROCO:

- WRF (Weather Research and Forecast model developed at NCAR, Boulder, USA)
- Meso-NH (Mesoscale Non-Hydrostatic model developed at Laboratoire d'Aérologie, Toulouse, France)
- WW3 (WaveWatch III model developed at NCEP, USA, and Ifremer, France)
- ...

Those model are not provided for download with CROCO and need to be installed separately, as well as OASIS-MCT library.

A description of the OASIS-MCT features, its implementation in CROCO, WW3 and WRF codes, and the coupled variables that can be exchanged are given in the following.

Datailed step by step coupled tutorial is also available in the **Tutorials** section.

16.1 OASIS philosophy

16.1.1 OASIS libraries

OASIS-MCT libraries are:

- **psmile** for coupling
- **mct** (Argonne National Laboratory) for parallel exchanges
- **scrip** (Los Alamos National Laboratory) for interpolations

Functions provided by the OASIS-MCT framework are:

> **Note:** `oasis_` / `prism_` are new / old names for backward compatibility, both useable

- Initialization and creation of a local communicator for internal parallel computation in each model:
  - oasis_init_comp / prism_init_comp_proto
  - oasis_get_localcomm / prism_get_localcomm_proto
- Grid data definition for exchanges and interpolations:

  - oasis_write_grid
  - oasis_write_corner
  - oasis_write_area
  - oasis_write_mask
  - oasis_terminate_grids_writing
- Partition and exchanged variables definition:
  - oasis_def_partition / prism_def_partition_proto
  - oasis_def_var / prism_def_var_proto
  - oasis_enddef / prism_enddef_proto
- Exchange of coupling fields:
  - oasis_get / prism_get_proto
  - oasis_put / prism_put_proto
- Finalization:
  - oasis_terminate / prism_terminate_proto

These OASIS3-MCT intrinsic functions are called in each model involved in the coupling. **Initialization** phase, **Definition** phase, and **Finalization** phase are called only once in each simulation while **Exchange** phase is called every time step. The effective exchanges are done only at specified times, defined by the coupling frequency, although the **Exchange** phase is called every model time step. The coupling frequency is controlled through the OASIS3-MCT namcouple.

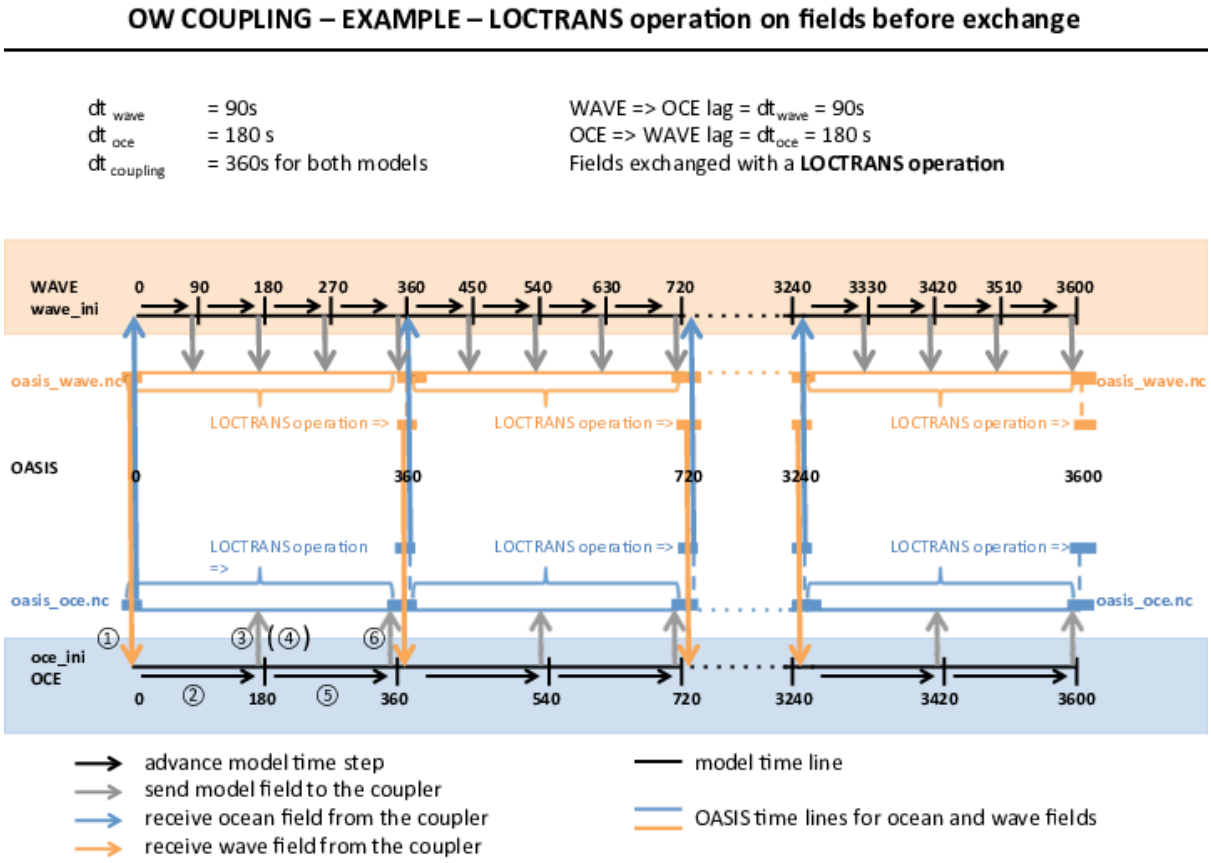
16.1.2 Coupling sequence

The frequency of exchanges between two models is defined by the **coupling time step**.

The **coupling time step** must be a multiple of the models time steps. An example of coupling sequence is pictured in the following Figure. In this example, the coupling time step is defined at 360s for both models. The wave model time step is 90s, so it will exchange every 4 time steps. The ocean model time step is 180s, so it will exchange every 2 time steps.

Another coupling parameter defined in the namcouple is the **lag**. It is used by the OASIS coupler to synchronize the `send` and `receive` functions. The lag must be defined for each model at the same value than its own time step. For instance:

- WAVE to OCEAN `lag = dt wave = 90`
- OCEAN to WAVE `lag = dt ocean = 180`



Therefore, `receive` and `send` functions have to be set at the same time in the model codes. OASIS will send the fields at the appropriate time thanks to the lag defined in the namcouple.

The coupling sequence in each model is:

initialization	`oasis_time = 0`
reception of coupled fields	`rcv(oasis_time)`
model time stepping	computation `t -> t+dt`
sending of coupled fields	`snd(oasis_time)`
increment of coupling time	`oasis_time = oasis_time + dt`

OASIS will exchange fields (get/put) if the time corresponds to a coupling time step, *e.g.* if:

- `oasis_time` corresponds to a coupling time step for get
- `oasis_time + lag` corresponds to a coupling time step for put

IN THE MODEL	IN OASIS
receive (date)	get(date)
send(date)	put(date+lag)

OASIS is also able to store fields from a model if a time transformation is requested in the namcouple (keyword `LOCTRANS + type of transformation`, see next section). OASIS will store the fields until a coupling time step is reached, then it will apply the time transformation, interpolate spatially the field as specified in the namcouple, and exchange the field with the other model.

16.1.3 Restart files

As reception of coupled fields is called before model computation, you need to create **restart files** for the coupler containing initial or restart fields for the first time step.

These **restart files** are for OASIS, and therefore need to have variable names corresponding to OASIS `namcouple` coupled fields. The initial files for OASIS are named `oasis_oce.nc` and `oasis_wave.nc` in the example pirctured in the above Figure. `oce_ini` and `wave_ini` are not related to OASIS, they are usual initialization or restart files from your oceanic and wave model; *e.g.* in CROCO, `oce_ini` is `croco_ini.nc`, and in WW3, `wave_ini` is `restart.ww3`).

Summary of the restart files:

- `oasis_oce.nc`, `oasis_wave.nc`: restart files for OASIS, you need to create them at the beginning of the run, OASIS will overwrite them at the end of the run, and they will be available for next restart
- `oce_ini`, `wave_ini`: correspond to `croco_ini.nc`, `restart.ww3`. These are your ocean and wave model initial or restart files

Practical example of the coupling sequence pictured in the above Figure:

```
oasis_time = 0
#1 => get field from oasis_wave.nc
rcv(0) => in oasis: get(0)
#2 => timestepping
t = 0+dt = 0+180 = 180
#3 => 180 is not a coupling time step, do nothing
snd(0) => in oasis: put(0+lag) = put(0+180) = put(180)
oasis_time = oasis_time+dt = 0+180 = 180
#4 => 180 is not a coupling time step, do nothing
rcv(180) => in oasis: get(180)
#5 => timestepping
t = 180+dt = 180+180 = 360
#6 => 360 is a coupling time step, put field
snd(180) => in oasis: put(180+lag) = put(180+180) = put(360)
```

16.1.4 Interpolations

The OASIS3-MCT coupler can process time transformations and 2D spatial interpolations of the exchanged fields. The 2D spatial interpolation, requested if models have different grids, is performed by the **scrip** library using `SCRIPR` keyword in the `namcouple`. Available interpolation types are:

`BILINEAR`	interpolation based on a local bilinear approximation
`BICUBIC`	interpolation based on a local bicubic approximation
`CONSERV`	1st or 2nd order conservative remapping
`DISTWGT`	distance weighted nearest-neighbour interpolation (N neighbours)
`GAUSWGT`	N nearest-neighbour interpolation weighted by their distance and a gaussian function

See OASIS manual for detailed informations.

Time transformations can also be performed by OASIS using `LOCTRANS` keywork in the `namcouple`. Available transformations are:

INSTANT	no time transformation, the instantaneous field is transfered
ACCUMUL	the field accumulated over the previous coupling period is exchanged
AVERAGE	the field averaged over the previous coupling period is transfered
T_MIN	the minimum value of the field for each source grid point over the previous coupling period is transfered
T_MAX	the maximum value of the field for each source grid point over the previous coupling period is transfered

16.2 Detailed OASIS implementation

16.2.1 In CROCO

The following routines are specifically built for coupling with OASIS and contain calls to OASIS intrinsic functions:

- `cpl_prism_init.F`: Manage the initialization phase of OASIS3-MCT : local MPI communicator
- `cpl_prism_define.F`: Manage the definition phase of OASIS3-MCT: domain partition, name of exchanged fields as read in the namcouple
- `cpl_prism_grid.F`: Manage the definition of grids for the coupler
- `cpl_prism_put.F`: Manage the sending of arrays from CROCO to the OASIS3-MCT coupler
- `cpl_prism_getvar.F`: Manage the generic reception from OASIS3-MCT.
- `cpl_prism_get.F`: Manage the specificity of each received variable: C-grid position, and field unit transformations

These routines are called in the the code in:

- `main.F`: Initialization, and finalization phases
- `get_initial.F`: Definition phase
- `zoom.F`: Initialization phase for AGRIF nested simulations
- `step.F`: Exchanges (sending and reception) of coupling variables

Other CROCO routines have also been slightly modified to introduce coupling:

- `testkeys.F`: To enable automatic linking to OASIS3-MCT libraries during compilation with jobcomp
- `cppdefs.h`: Definition of the `OA_COUPLING` and `OW_COUPLING` cpp-keys, and the other related and requested cpp-keys, as `MPI`
- `set_global_definitions.h`: Definition of cpp-keys in case of coupling (`undef OPENMP`, `define MPI`, `define MPI_COMM_WORLD ocean_grid_comm`: MPI_COMM_WORLD generic MPI communicator is redefined as the local MPI communicator ocean_grid_comm, `undef BULK_FLUX`: no bulk OA parametrization)

- `mpi_roms.h`: Newly added to define variables related to OASIS3-MCT operations. It manage the MPI communicator, using either the generic MPI_COMM_WORLD, either the local MPI communicator created by OASIS3-MCT
- `read_inp.F`: Not reading atmospheric forcing files (`croco_frc.nc` and/or `croco_blk.nc`) in OA coupled mode

A schematic picture of the calls in CROCO is (with # name.F indicating the routine we enter in):

```
# main.F
if !defined AGRIF
call cpl_prism_init
else
call Agrif_MPI_Init
endif
...
call read_inp
...
call_get_initial
       # get_initial.F
       ...
       call cpl_prism_define
               # cpl_prism_define.F
               call prism_def_partition_proto
               call cpl_prism_grid
               call prism_def_var_proto
               call prism_enddef_proto
       oasis_time=0
# main.F
...
DO 1:NT
call step
       # step.F
       if ( (iif==-1).and.(oasis_time>=0).and.(nbstep3d<ntimes) ) then
               call cpl_prism_get(oasis_time)
                       # cpl_prism_get.F
                       call cpl_prism_getvar
       endif
       call prestep3d
               call get_vbc
               ...
       call step2d
       ...
       call step3d_uv
       call step3d_t
       iif = -1
       nbstep3d = nbstep3d + 1
       if (iif==-1) then
                if (oasis_time>=0.and.(nbstep3d<ntimes)) then
                       call cpl_prism_put (oasis_time)
                       oasis_time = oasis_time + dt
               endif
       endif
# main.F
END DO
...
call prism_terminate_proto
...
```

16.2.2 In WW3

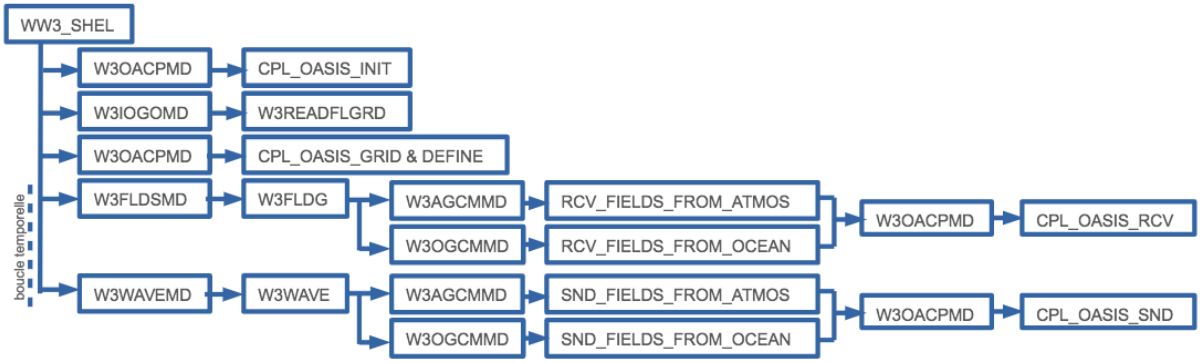
The following routines have been specifically built for coupling with OASIS:

- `w3oacpmd.ftn`: main coupling module with calls to oasis intrinsic functions
- `w3agcmmd.ftn`: module for coupling with an atmospheric model
- `w3ogcmmd.ftn`: module for coupling with an ocean model

The following routines have been modified for coupling with OASIS:

- `w3fldsmd.ftn`: routine that manage input fields, and therefore received fields from the coupler
- `w3wdatmd.ftn`: routine that manage data structure for wave model, and therefore time for coupling
- `w3wavemd.ftn`: actual wave model, here is located the sending of coupled variables
- `ww3_shel.ftn`: main routine managing the wave model, definition/initialisation/partition phases are located here

A schematic picture of the calls in WW3 is given here:



16.2.3 In WRF

The routines specifically built for coupling are:

- `module_cpl_oasis3.F`
- `module_cpl.F`

Implementation of coupling with the ocean implies modifications in the following routines:

- `phys/module_bl_mynn.F`
- `phys/module_bl_ysu.F`
- `phys/module_pbl_driver.F`
- `phys/module_surface_driver.F`
- `phys/module_sf_sfclay.F`
- `phys/module_sf_sfclayrev.F`

Implementation of coupling with waves implies modifications in the following routines:

- `Regristry/Registry.EM_COMMON`: CHA_COEF added
- `dyn/module_first_rk_step_part1.F`: CHA_COEF=grid%cha_coef declaration added
- `frame/module_cpl.F`: rcv CHA_COEF added
- `phys/module_sf_sfclay.F` and `..._sfclayrev.F`: introduction of wave coupled case: isftcflx=5 as follows:

```
! SJ: change charnock coefficient as a function of waves, and hence roughness
! length
  IF ( ISFTCFLX.EQ.5 ) THEN
    ZNT(I)=CHA_COEF(I)*UST(I)*UST(I)/G+0.11*1.5E-5/UST(I)
  ENDIF
```

- `phys/module_surface_driver.F`: CHA_COEF added in calls to sfclay and sfclayrev and "CALL cpl_rcv" for CHA_COEF

Schematic picture of WRF architecture and calls to the coupling dependencies:

```
# main/wrf.F
CALL wrf_init

   # main/module_wrf_top.F
   CALL wrf_dm_initialize

      # frame/module_dm.F
      CALL cpl_init( mpi_comm_here )
      CALL cpl_abort( 'wrf_abort', 'look for abort message in rsl* files' )

   CALL cpl_defdomain( head_grid )

# main/wrf.F
CALL wrf_run

   # main/module_wrf_top.F
   CALL integrate ( head_grid )

      # frame/module_integrate.F
      CALL cpl_defdomain( new_nest )
      CALL solve_interface ( grid_ptr )

         # share/solve_interface.F
         CALL solve_em ( grid , config_flags ... )

            # dyn_em/solve_em.F
            curr_secs2  # time for the coupler
            CALL cpl_store_input( grid, config_flags )
            CALL cpl_settime( curr_secs2 )
            CALL first_rk_step_part1

               # dyn_em/module_first_rk_step_part1.F
               CALL surface_driver( ... )

                  # phys/module_surface_driver.F
                  CALL cpl_rcv( id, ... )
                  u_phytmp(i,kts,j)=u_phytmp(i,kts,j)-uoce(i,j)
                  v_phytmp(i,kts,j)=v_phytmp(i,kts,j)-voce(i,j)

                  CALL SFCLAY( ... cha_coef ...)
                     # phys/module_sf_sfclay.F
                     CALL SFCLAY1D
                     IF ( ISFTCFLX.EQ.5 ) THEN
                       ZNT(I)=CHA_COEF(I)*UST(I)*UST(I)/G+0.11*1.5E-5/UST(I)
                     ENDIF

                  CALL SFCLAYREV( ... cha_coef ...)
                     # phys/module\_sf\_sfclayrev.F
                     CALL SFCLAYREV1D
                     IF ( ISFTCFLX.EQ.5 ) THEN
                       ZNT(I)=CHA_COEF(I)*UST(I)*UST(I)/G+0.11*1.5E-5/UST(I)
```

(continues on next page)

(continued from previous page)

```
                    ENDIF

                # dyn_em/module_first_rk_step_part1.F
                CALL pbl_driver( ... )

                   # phys/module_pbl_driver.F
                   CALL ysu( ... uoce,voce, ... )
                      # module_bl_ysu.F
                      call ysu2d ( ... uox,vox, ...)
                      wspd1(i) = sqrt( (ux(i,1)-uox(i))*(ux(i,1)-uox(i))
                                     + (vx(i,1)-vox(i))*(vx(i,1)-vox(i)) )+1.e-9
                      f1(i,1) = ux(i,1)+uox(i)*ust(i)**2*g/del(i,1)*dt2/wspd1(i)
                      f2(i,1) = vx(i,1)+vox(i)*ust(i)**2*g/del(i,1)*dt2/wspd1(i)

                   CALL mynn_bl_driver( ... uoce,voce, ... )
                      # module_bl_mynn.F
                      d(1)=u(k)+dtz(k)*uoce*ust**2/wspd
                      d(1)=v(k)+dtz(k)*voce*ust**2/wspd

             # dyn_em/solve_em.F
             CALL first_rk_step_part2

       # frame/module_integrate.F
       CALL cpl_snd( grid_ptr )

   # Check where this routine is called...
   # frame/module_io_quilt.F  # for IO server (used with namelist variable: nio_
↪tasks_per_group
   CALL cpl_set_dm_communicator( mpi_comm_local )
   CALL cpl_finalize()

# main/wrf.F
CALL wrf_finalize
    #main/module_wrf_top.F
    CALL cpl_finalize()
```

16.3 Coupled variables

16.3.1 Coupling with an atmospheric model

When coupling CROCO to an atmospheric model, to have a consistent interface, you should use momentum and heat fluxes computed from the atmospheric model bulk formula.

No surface forcing file is required (only boundary forcing, and eventually tidal forcing).

The following cpp-keys have to be set:

```
# define OA_COUPLING
# define MPI

# undef  BULK_FLUX
# undef  SMFLUX_CFB
```

Note: `SMFLUX_CFB` is a cpp-key to use a wind stress relative to the current in forced mode. In coupled mode, as current is sent to the atmosphere, the wind stress from the atmospheric model account for such a current feedback.

Fields sent by CROCO		
Name (units)	name and eventual oper. in the model	OASIS name
SST (K)	t(:,:,N,nnew,itemp) + 273.15	CROCO_SST
U-component of current (m/s)	u (at rho points): 0.5*(u(1:Lmmpi,1:Mmmpi,N,nnew) +u(2:Lmmpi+1,1:Mmmpi,N,nnew))	CROCO_UOCE
V-component of current (m/s)	v (at rho points): 0.5*(v(1:Lmmpi,1:Mmmpi,N,nnew) +v(1:Lmmpi,2:Mmmpi+1,N,nnew))	CROCO_VOCE
Eastward component of current (m/s)	u (at rho points) rotated eastwards (useful for rotated grids) (0.5 * (u(1:Lmmpi ,1:Mmmpi,N,nnew) + u(2:Lmmpi+1,1:Mmmpi,N,nnew))) * cos(angler(1:Lmmpi ,1:Mmmpi)) - (0.5 * (v(1:Lmmpi,1:Mmmpi ,N,nnew) + v(1:Lmmpi,2:Mmmpi+1,N,nnew))) * sin(angler(1:Lmmpi ,1:Mmmpi)) +u(2:Lmmpi+1,1:Mmmpi,N,nnew))	CROCO_EOCE
Northward component of current (m/s)	v (at rho points) rotated northward (useful for rotated grids) (0.5 * (u(1:Lmmpi ,1:Mmmpi,N,nnew) + u(2:Lmmpi+1,1:Mmmpi,N,nnew))) * sin(angler(1:Lmmpi ,1:Mmmpi)) + (0.5 * (v(1:Lmmpi,1:Mmmpi ,N,nnew) + v(1:Lmmpi,2:Mmmpi+1,N,nnew))) * cos(angler(1:Lmmpi ,1:Mmmpi))	CROCO_NOCE

Fields received by CROCO		
Name (units)	name and eventual oper. in the model	OASIS name
U component of wind stress (N/m2)	sustr (at u point): 0.5*(FIELD(io-1,jo)+FIELD(io,jo))/rho0 if eastward, it is first rotated: FIELD = etau * cos(angler) + ntau * sin(angler)	CROCO_UTAW or CROCO_ETAW
V component of wind stress (N/m2)	svstr (at v point): 0.5*(FIELD(io,jo-1)+FIELD(io,jo))/rho if northward, it is first rotated: FIELD = ntau * cos(angler) - etau * sin(angler)	CROCO_VTAW or CROCO_NTAW
Wind stress module (N/m2)	smstr = FIELD / rho0	CROCO_TAUM
Surface net solar flux (W/m2)	srflx = FIELD / (rho0*Cp)	CROCO_SRFL
Surface net non-solar flux (W/m2)	stflx(:,:,itemp) = FIELD / (rho0*Cp)	CROCO_STFL
Evaporation-Precipitation (kg/m2/s)	stflx(:,:,isalt) = FIELD / 1000	CROCO_EVPR
Surface atmospheric pressure (Pa)	patm2d = FIELD	CROCO_PSFC

Fields received by WRF		
Name (units)	name (in the model)	OASIS name
SST (K)	SST	WRF_d01_EXT_d01_SST
U component of current (m/s)	UOCE	WRF_d01_EXT_d01_UOCE
V component of current (m/s)	VOCE	WRF_d01_EXT_d01_VOCE
Eastward component of current (m/s)	EOCE	WRF_d01_EXT_d01_EOCE
Northward component of current (m/s)	NOCE	WRF_d01_EXT_d01_NOCE

Fields sent by WRF		
Name (units)	name (in the model)	OASIS name
Surface net solar flux (W/m2)	GSW	WRF_d01_EXT_d01_SURF_NET_SOLAR
Surface net non-solar flux (W/m2)	GLW-STBOLT*EMISS*SST**4-LH-HFX	WRF_d01_EXT_d01_SURF_NET_NON-SOLAR
Evaporation-precipitation (kg/m2/s)	QFX-(RAINCV+RAINNCV)/DT	WRF_d01_EXT_d01_EVAP-PRECIP
Surface atmospheric pressure (Pa)	PSFC	WRF_d01_EXT_d01_PSFC
Wind stress module (N/m2)	taut = rho * ust**2	WRF_d01_EXT_d01_TAUMOD
U component of wind stress (N/m2)	taui = taut * u_uo / wspd	WRF_d01_EXT_d01_TAUX
V component of wind stress (N/m2)	tauj = taut * v_uo / wspd	WRF_d01_EXT_d01_TAUY
Eastward comp. of wind stress(N/m2)	cosa * taui - sina * tauj	WRF_d01_EXT_d01_TAUE
Northward comp. of wind stress(N/m2)	cosa * tauj + sina * taui	WRF_d01_EXT_d01_TAUN

Note: If you decide to couple CROCO with multiple WRF domains, variables coming from WRF will be defined by adding _EXT*. Here * corresponds to which domains the variable is coming (1=Parent, 2=Nest 1 ,. . .).

16.3.2 Coupling with a wave model

When coupling CROCO to a wave model, the wave-current interactions have to be set on. At the moment, only mean wave parameters are exchanged, their contribution to ocean dynamics is computed into the wave-current interaction routine in CROCO.

The following cpp-keys have to be set:

```
# define OW_COUPLING
# define MPI

# define MRL_WCI
```

Note: You also have to be careful to the choice of the momentum flux. For better consistency, here we suggest to account for the momentum flux seen by the wave model, and thus set:

```
# undef  BULK_FLUX
# define WAVE_SMFLUX
```

Fields sent by CROCO		
Name (units)	name and eventual oper. in the model	OASIS name
SSH (m)	zeta	CROCO_SSH
U-component of current (m/s)	u (at rho points): 0.5*(u(1:Lmmpi,1:Mmmpi,N,nnew) +u(2:Lmmpi+1,1:Mmmpi,N,nnew))	CROCO_UOCE
V-component of current (m/s)	v (at rho points): 0.5*(v(1:Lmmpi,1:Mmmpi,N,nnew) +v(1:Lmmpi,2:Mmmpi+1,N,nnew))	CROCO_VOCE
Eastward component of current (m/s)	u (at rho points) rotated to east (useful for rotated grids) (0.5 * (u(1:Lmmpi ,1:Mmmpi,N,nnew) + u(2:Lmmpi+1,1:Mmmpi,N,nnew))) * cos(angler(1:Lmmpi ,1:Mmmpi)) - (0.5 * (v(1:Lmmpi,1:Mmmpi ,N,nnew) + v(1:Lmmpi,2:Mmmpi+1,N,nnew))) * sin(angler(1:Lmmpi ,1:Mmmpi)) +u(2:Lmmpi+1,1:Mmmpi,N,nnew))	CROCO_EOCE
Northward component of current (m/s)	v (at rho points) rotated to north (useful for rotated grids) (0.5 * (u(1:Lmmpi ,1:Mmmpi,N,nnew) + u(2:Lmmpi+1,1:Mmmpi,N,nnew))) * sin(angler(1:Lmmpi ,1:Mmmpi)) + (0.5 * (v(1:Lmmpi,1:Mmmpi ,N,nnew) + v(1:Lmmpi,2:Mmmpi+1,N,nnew))) * cos(angler(1:Lmmpi ,1:Mmmpi))	CROCO_NOCE

Fields received by CROCO		
Name (units)	name and eventual oper. in the model	OASIS name
Significant wave height (m)	whrm = FIELD * 0.70710678	CROCO_HS
Mean wave period (s) -> frequency	wfrq = 2*pi / FIELD	CROCO_T0M1
Mean wave direction -> wavenumbers	wdrx = cos(FIELD - angler) wdre = sin(FIELD - angler)	CROCO_DIR
U component of wave stress (m2/s2)	twox (at u point): 0.5*(FIELD(io-1,jo)+FIELD(io,jo)) if eastward, it is first rotated: FIELD = etwo * cos(angler) + ntwo * sin(angler)	CROCO_UTWO or CROCO_ETWO
V component of wave stress (m2/s2)	twoy (at v point): 0.5*(FIELD(io,jo-1)+FIELD(io,jo)) if northward, it is first rotated: FIELD = ntwo * cos(angler) - etwo * sin(angler)	CROCO_VTWO or CROCO_NTWO
U comp. of wind-to-wave stress (m2/s2)	tawx (at u point): 0.5*(FIELD(io-1,jo)+FIELD(io,jo)) if eastward, it is first rotated: FIELD = etaw * cos(angler) + ntaw * sin(angler)	CROCO_UTAW or CROCO_ETAW
V comp. of wind-to-wave stress (m2/s2)	tawy (at v point): 0.5*(FIELD(io,jo-1)+FIELD(io,jo)) if northward, it is first rotated: FIELD = ntaw * cos(angler) - etaw * sin(angler)	CROCO_VTAW or CROCO_NTAW

Other optional fields enventually sent, if not, they are analytically computed in the MRL_WCI routine		
Bernoulli head pressure (N/m)	bhd	CROCO_BHD
Wave-to-ocean TKE flux (W/m2)	foc	CROCO_FOC
Mean wavelength (m)	wlm	CROCO_LM
Wave orbital bottom velocity (m/s)	ubr = sqrt(ubrx**2+ubry**2)	CROCO_UBRX and CROCO_UBRY
Stokes drift surface velocity (m/s)	ust_ext = sqrt(ustx_ext**2+usty_ext**2)	CROCO_USSX and CROCO_USSY

Fields received by WW3		
Name (units)	name (in the model)	OASIS name
SSH = water level (m)	LEV	WW3__SSH
Zonal current (m/s)	CUR	WW3_OSSU
Meridional current (m/s)	CUR	WW3_OSSV

Fields sent by WW3		
Name (units)	name (in the model)	OASIS name
Mean wave period (s)	T0M1	WW3_T0M1
Significant wave height (m)	HS	WW3__OHS
Mean wave direction	THM	WW3__DIR
Zonal wave stress (N/m2)	TWOX	WW3_TWOX
Meridional wave stress (N/m2)	TWOY	WW3_TWOY
Zonal wind stress (N/m2)	TAWX	WW3_TAWX
Meridional wind stress(N/m2) TAWY WW3_TAWY		
Other fields possibly sent, but not used in coupling with CROCO at the moment		
Bernoulli head pressure (N/m)	BHD	WW3__BHD
Bottom orbital velocity (m/s)	UBR	WW3__UBR
Wave-to-ocean TKE flux (W/m2)	FOC	WW3__FOC
Mean wavelength (m)	LM	WW3___LM
Wave peak frequency (/s)	FP	WW3___FP

16.3.3 Coupling atmosphere and wave models

Fields received by WW3		
Name (units)	name (in the model)	OASIS name
Zonal wind (m/s)	WND	WW3__U10
Meridional wind (m/s)	WND	WW3__V10

Fields sent by WW3		
Name (units)	name (in the model)	OASIS name
Significant wave height (m)	HS	WW3__AHS
Charnock coefficient	ACHA	WW3_ACHA

Fields sent by WRF		
Name (units)	name (in the model)	OASIS name
Zonal wind at first level((m/s)	u_uo	WRF_d01_EXT_d01_WND_E_01
Meridional wind at first level (m/s)	v_vo	WRF_d01_EXT_d01_WND_N_01

Fields received by WRF		
Name (units)	name (in the model)	OASIS name
Charnock coefficient	CHA_COEF	WRF_d01_EXT_d01_CHA_COEF

16.3.4 Note on momentum flux when coupling 3 models

As the wave model has a quite complex parameterization of wave generation by winds, which is in subtle balance with the wave dissipation, the wind stress for the wave model is computed by its own parameterization. Therefore, to ensure energetic consistency of the momentum flux when coupling 3 models, we prescribe the wind stress in CROCO as:

```
sustr = sustr_from_atm_model - tawx + twox
svstr = svstr_from_atm_model - tawy + twoy

# where taw is stress from atm to waves
# and   two is stress from waves to ocean
```

16.3.5 Note on coupling with AGRIF

You may decide to coupled CROCO while using AGRIF. To do so, the variables sent by the parent domain (0) and the child domains (1,2,...) must be separated. Thus the variables sent, in case of using AGRIF, take the radical defined above (CROCO_VAR) to which we add _0 (for parent) or _1 (for first child). This gives, for example for variable SST, `CROCO_SST_0` or `CROCO_SST_1` for parent and child respectively.

For the variables received by CROCO, we will use its ability to handle CPLMASK. Each of the domains (parent or children) will be assigned a coupling mask named coupling_mask0.nc (parent), coupling_mask1.nc (child 1), each coupling mask being relative to its grid. The CROCO domain that receives a variable will be identified by its mask (CPLMASK*), which will be added to the previous radical. This will give CROCO_VAR_CPLMASK0 for the parent or CROCO_VAR_CPLMASK1 for child 1.

This makes it easy to define the received variables in a case where one decides to couple CROCO-AGRIF with several WRF domains. In this case the variables will have the nomenclature `CROCO_VAR_CPLMASK0` for the parent CROCO to which we add _EXT1 for the first domain of coupling_mask0.nc. By continuity `_EXT2` will correspond to domain 2 of coupling_mask0.nc. Then the variables received by CROCO, in a case of CROCO-AGRIF/WRF-nest simulation, will follow the format `CROCO_VAR_CPLMASK*_EXT*`.

16.4 Grids

16.4.1 OASIS grid files

OASIS manage grids and interpolations by using dedicated grid files:

- `grids.nc`
- `masks.nc`
- `areas.nc` (requested only for some of the interpolation types)

These files can be automatically created by OASIS functions called in each model, or can be created by the user in advance if specificities are requested. Some facilities are provided in `croco_tools/Coupling_tools` to create such grids.

If `grids.nc`, `masks.nc`, `areas.nc` exist in the working directory, they won't be overwritten by OASIS functions. So, be sure to have the good files or remove them before running the coupled model.

16.4.2 Multiple model grids (nesting case)

Multiple nested grids in the different models can be used in coupled mode.

The variables are therefore exchanged from/to the different grids. To do so, each coupled variable is identified in the coupler with its grid number:

- For CROCO the last character of the OASIS variable name defines the domain
  - 0 being the parent domain
  - 1 the first child domain, etc.
- For WRF the domains are defined by d01, d02, etc, and the target domain (CROCO for instance), by EXT_d01, EXT_d02, etc.

For example if you are coupling 2 CROCO domains to one atmospheric domain, you will specify 2 types of exchanges in the `namcouple`:

```
# exchange between CROCO parent domain to WRF domain
SRMSSTV0 WRF_d01_EXT_d01_SST

# exchange between CROCO child domain to WRF domain
SRMSSTV1 WRF_d01_EXT_d02_SST
```

If you are coupling 2 WRF domains to one CROCO domain:

```
# exchange between WRF d01 domain to CROCO domain
WRF_d01_EXT_d01_TAUMOD RRMTAUM0

# exchange between WRF d02 domain to CROCO domain
WRF_d02_EXT_d01_TAUMOD RRMTAUM0
```

Related CPP options:

OW_COUPLING	Activate Ocean-Wave coupling
OA_COUPLING	Activate Ocean-Atmosphere coupling
OA_MCT	Use OASIS-MCT for coupling

Preselected options:

```
#undef OA_COUPLING
#undef OW_COUPLING
```

CHAPTER

SEVENTEEN

I/O AND ONLINE DIAGNOSTICS

Basic CPP options:

AVERAGES	Process and output time-averaged data
AVERAGES_K	Process and output time-averaged vertical mixing
XIOS	Use XIOS IO server (only version >= 2 is supported)

XIOS is an external library for output (developed at IPSL) providing for flexibility and design to improve performances for HPC : see http://forge.ipsl.jussieu.fr/ioserver

Preselected options:

```
# define AVERAGES
# define AVERAGES_K
# undef XIOS
```

Advanced diagnostics CPP options:

DIAGNOSTICS_TS	Store and output budget terms of the tracer equations
DIAGNOSTICS_TS_ADV	Choose advection rather than transport formulation for tracer budgets
DIAGNOSTICS_TS_MLD	Integrate tracer budgets over the mixed-layer depth
DIAGNOSTICS_TSVAR	Store and output budget terms of the tracer variance equations (instead of tracer)
DIAGNOSTICS_UV	Store and output budget terms of the momentum equations.
DIAGNOSTICS_BARO	Isolate contribution from barotropic/baroclinic coupling (included in the vertical mixing term otherwise) for momentum, barotropic vorticity and kinetic energy budgets
DIAGNOSTICS_VRT	Store and output budget terms of the barotropic vorticity equation
DIAGNOSTICS_EK	Store and output budget terms of the kinetic energy equation (vertically integrated)
DIAGNOSTICS_EDDY	Store and output time-averaged quadratic quantities u^2, v^2, u*v, u*w, v*w, u*b, v*b, w*b, u*sustr, v*svstr, u*bustr, v*bvstr, zeta^2
DIAGNOSTICS_PV	Store and output non conservative term in the momentum equations and diabatic term in the tracer equations. if DIAGNOSTICS_DISS is also defined, terms are multiplied by momentum and thermal/saline expension coefficients to be used to estimate kinetic and potential energy dissipation.

The different budgets and their computation are detailled in https://www.jgula.fr/Croco/diagnostics_croco.pdf

Preselected options:

```
# undef DIAGNOSTICS_TS     Store and output budget terms of the tracer equations
# undef DIAGNOSTICS_TS_ADV Choose advection rather than transport formulation for␣
↪tracer budgets
# undef DIAGNOSTICS_TS_MLD Integrate tracer budgets over the mixed-layer depth
# undef DIAGNOSTICS_TSVAR  output budgets of tracer variance instead of tracer
# undef DIAGNOSTICS_UV     Store and output budget terms of the momentum equations
# undef DIAGNOSTICS_BARO   Isolate contribution from barotropic/baroclinic coupling
# undef DIAGNOSTICS_VRT    Store and output budget terms of the barotropic␣
↪vorticity equation
# undef DIAGNOSTICS_EK     Store and output budget terms of the kinetic energy␣
↪equation (vertically integrated)
# undef DIAGNOSTICS_PV     Store and output non conservative / diabatic terms
# undef DIAGNOSTICS_EDDY   Store and output time-averaged quadratic quantities
```

CHAPTER

EIGHTEEN

REVIEW OF TEST CASES

18.1 Basin

This is a rectangular, flat-bottomed basin with double-gyre wind forcing. It produces a western boundary current flowing into a central Gulf Stream which goes unstable and generates eddies if resolution is increased

```
# define BASIN
```

CPP options:

```
# undef  OPENMP
# undef  MPI
# define UV_ADV
# define UV_COR
# define UV_VIS2
# define SOLVE3D
# define TS_DIF2
# define BODYFORCE
# define ANA_GRID
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_STFLUX
# define ANA_BTFLUX
# define NO_FRCFILE
```

Settings :

Results :

18.2 Canyon

```
# define CANYON
```

CPP options:

```
# undef  OPENMP
# undef  MPI
# define CANYON_STRAT
# define UV_ADV
# define UV_COR
# define SOLVE3D
# define EW_PERIODIC
# define ANA_GRID
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_STFLUX
```

(continues on next page)

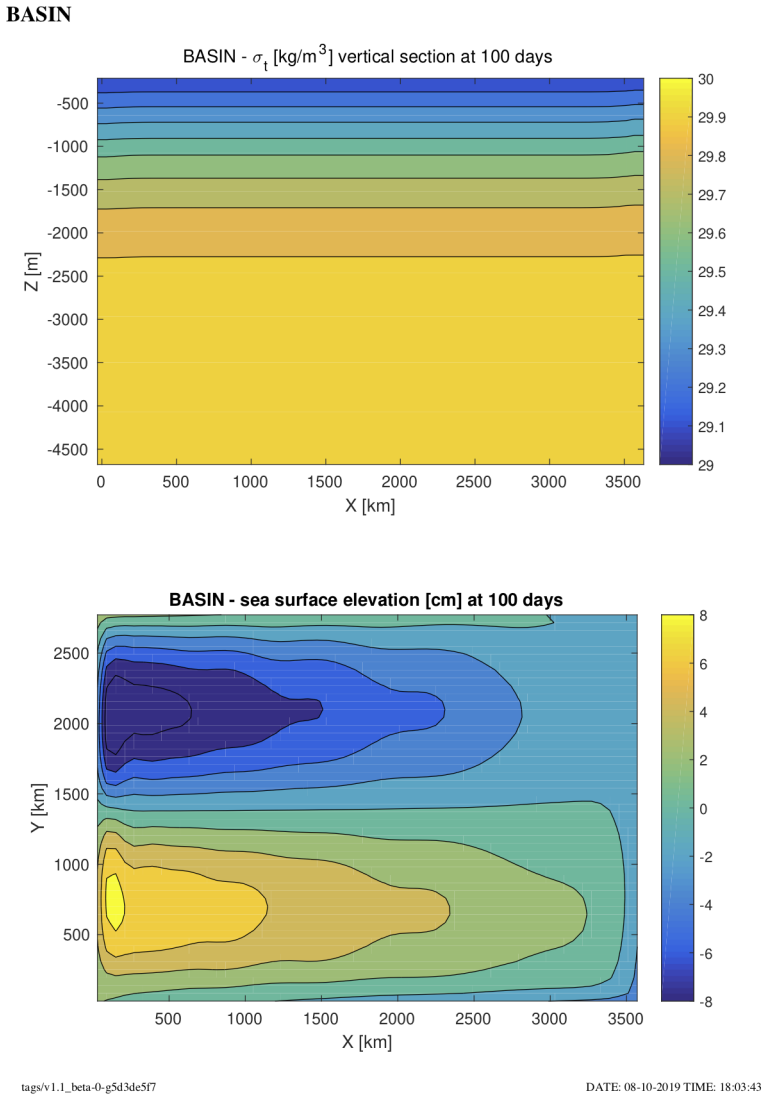


Fig. 1: BASIN results : density (up) and sea surface elevation (down)

(continued from previous page)

```
# define ANA_BTFLUX
# define NO_FRCFILE
```

Settings :

Results :

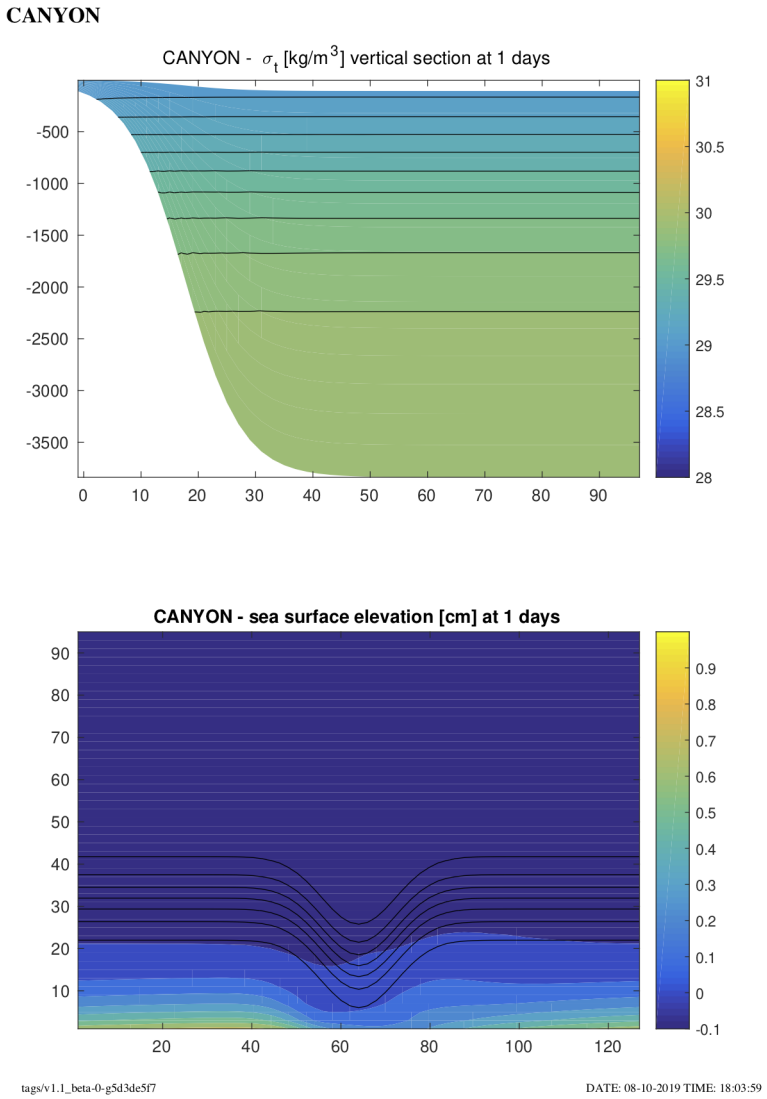


Fig. 2: CANYON results : density (up) and sea surface elevation (down)

18.3 Equator

Boccaletti, G., R.C. Pacanowski, G.H. Philander and A.V. Fedorov, 2004, The Thermal Structure of the Upper Ocean, J.Phys.Oceanogr., 34, 888-902.

```
# define EQUATOR
```

CPP options:

```
# undef  OPENMP
# undef  MPI
# define UV_ADV
# define UV_COR
# define UV_VIS2
# define SOLVE3D
# define TS_DIF2
# define ANA_GRID
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_STFLUX
# define ANA_SRFLUX
# define ANA_SSFLUX
# define ANA_BTFLUX
# define ANA_BSFLUX
# define QCORRECTION
# define ANA_SST
# define LMD_MIXING
# define LMD_RIMIX
# define LMD_CONVEC
# define NO_FRCFILE
```

Settings :

Results :

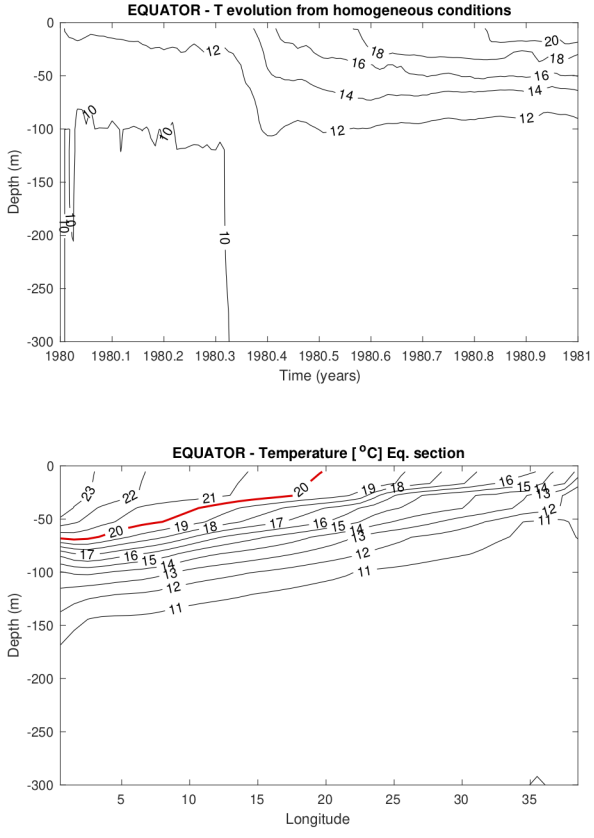


Fig. 3: EQUATOR results : temperature , time evolution (up) vertical section (down)

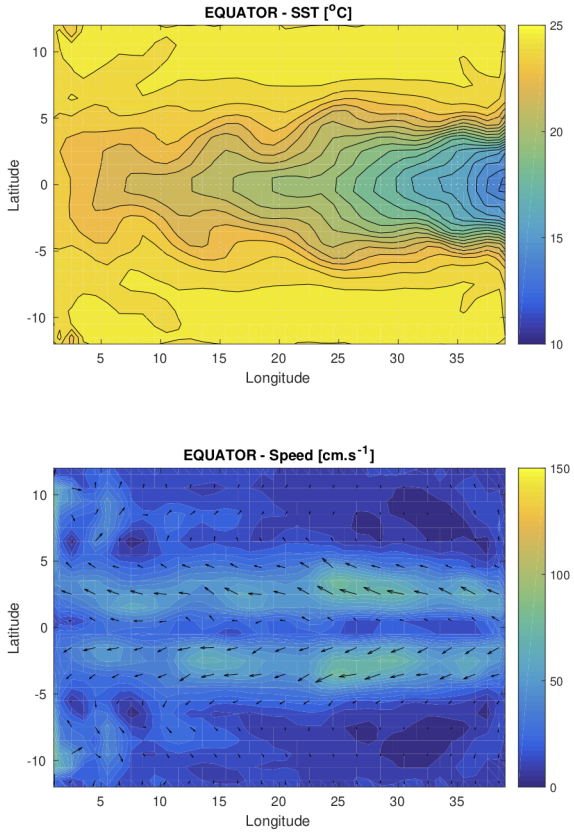


Fig. 4: EQUATOR results : speed (up) and sea surface elevation (down)

18.4 Inner Shelf

Compare wind driven innershelf dynamics between 2D Ekman theory and CROCO numerical solution

Estrade P., P. Marchesiello, A. Colin de Verdiere, C. Roy, 2008: Cross-shelf structure of coastal upwelling : a two-dimensional expansion of Ekman's theory and a mechanism for innershelf upwelling shut down. Journal of Marine Research, 66, 589-616.

Marchesiello P., and P. Estrade, 2010: Upwelling limitation by geostrophic onshore flow. Journal of Marine Research, 68, 37-62.

```
# define INNERSHELF
```

CPP options:

```
# undef  OPENMP
# undef  MPI
# undef  NBQ
# define INNERSHELF_EKMAN
# define INNERSHELF_APG
# define SOLVE3D
# define UV_COR
# define ANA_GRID
# define ANA_INITIAL
# define AVERAGES
# define ANA_SSFLUX
# define ANA_SRFLUX
# define ANA_STFLUX
# define ANA_BSFLUX
# define ANA_BTFLUX
```

(continues on next page)

(continued from previous page)

```
# define ANA_SMFLUX
# define NS_PERIODIC
# define OBC_WEST
# define SPONGE
# ifndef INNERSHELF_EKMAN
#  define UV_ADV
#  define SALINITY
#  define NONLIN_EOS
#  define LMD_MIXING
#  undef  GLS_MIXING
#  ifdef LMD_MIXING
#   define LMD_SKPP
#   define LMD_BKPP
#   define LMD_RIMIX
#   define LMD_CONVEC
#  endif
#  undef WAVE_MAKER_INTERNAL
#  ifdef WAVE_MAKER_INTERNAL
#   define ANA_BRY
#   define Z_FRC_BRY
#   define M2_FRC_BRY
#   define M3_FRC_BRY
#   define T_FRC_BRY
#  endif
# endif
# define NO_FRCFILE
```

Settings :

Results :

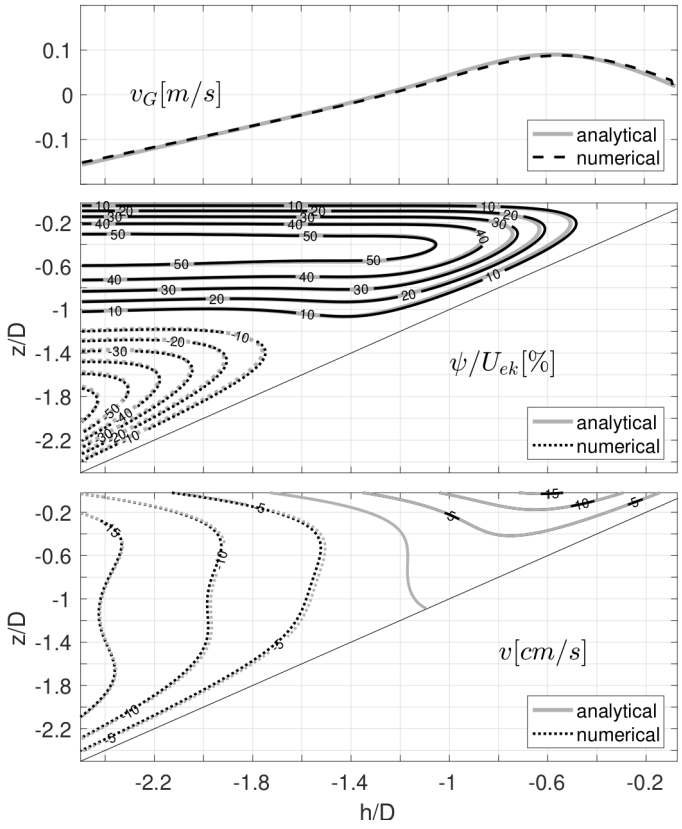


Fig. 5: INNERSHELF results : comparison with analytical solution

18.5 River Runoff

```
# define RIVER
```

CPP options:

```
# undef  OPENMP
# undef  MPI
# define SOLVE3D
# define UV_ADV
# define UV_COR
# define NONLIN_EOS
# define SALINITY
# define ANA_GRID
# define MASKING
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_STFLUX
# define ANA_SSFLUX
# define ANA_SRFLUX
# define ANA_BTFLUX
# define ANA_BSFLUX
# define LMD_MIXING
# define LMD_SKPP
# define LMD_BKPP
# define LMD_RIMIX
# define LMD_CONVEC
# define PSOURCE
# define ANA_PSOURCE
# define NS_PERIODIC
# undef  FLOATS
# ifdef FLOATS
#  define RANDOM_WALK
#  ifdef RANDOM_WALK
#   define DIEL_MIGRATION
#   define RANDOM_VERTICAL
#   define RANDOM_HORIZONTAL
#  endif
# endif
# define NO_FRCFILE
```

Settings :

Results :

18.6 Gravitational/Overflow

```
# define OVERFLOW
```

CPP options:

```
# undef  OPENMP
# undef  MPI
# define UV_ADV
# define UV_COR
# define UV_VIS2
# define TS_DIF2
# define TS_MIX_GEO
# define SOLVE3D
```

(continues on next page)

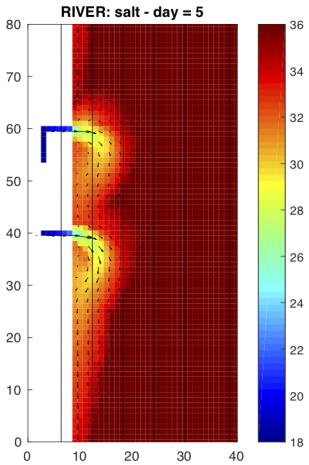


Fig. 6: RIVER results : river plume

(continued from previous page)

```
# define ANA_GRID
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_STFLUX
# define ANA_BTFLUX
# define NO_FRCFILE
```

Setting :

Results :

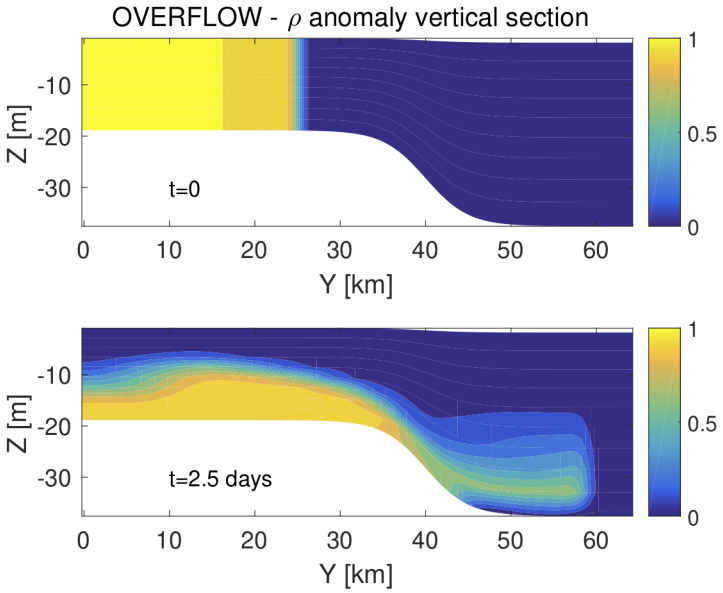


Fig. 7: OVERFLOW results : initial state (up) and density evolution (down)

18.7 Seamount

```
# define SEAMOUNT
```

CPP options:

```
# undef  OPENMP
# undef  MPI
# define UV_ADV
# define UV_COR
# define SOLVE3D
# define SALINITY
# define NONLIN_EOS
# define ANA_GRID
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_STFLUX
# define ANA_SSFLUX
# define ANA_SRFLUX
# define ANA_BTFLUX
# define ANA_BSFLUX
# define NO_FRCFILE
```

Settings :

Results :

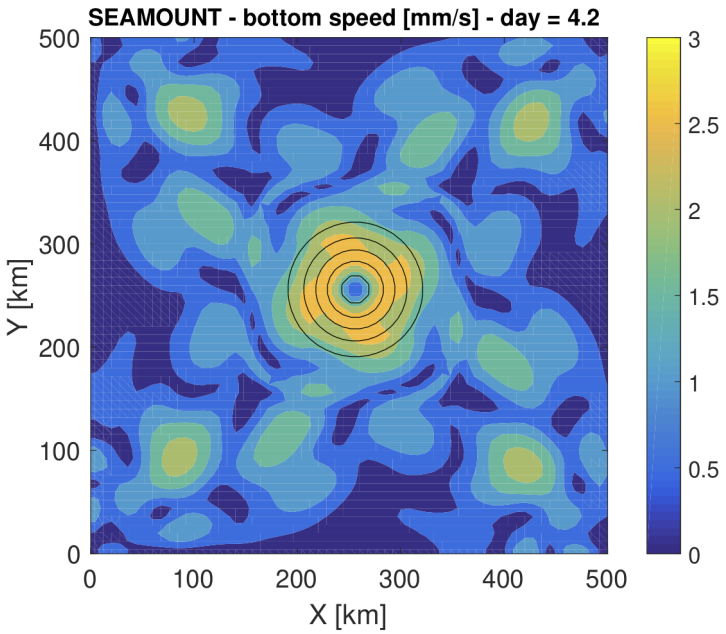


Fig. 8: SEAMOUNT results : bottom speed

18.8 Shelf front

```
# define SHELFRONT
```

CPP options:

```
# undef  OPENMP
# undef  MPI
# define UV_ADV
# define UV_COR
# define SOLVE3D
# define SALINITY
# define ANA_GRID
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_STFLUX
# define ANA_SSFLUX
# define ANA_SRFLUX
# define ANA_BTFLUX
# define ANA_BSFLUX
# define EW_PERIODIC
# define NO_FRCFILE
```

Settings :

Results :

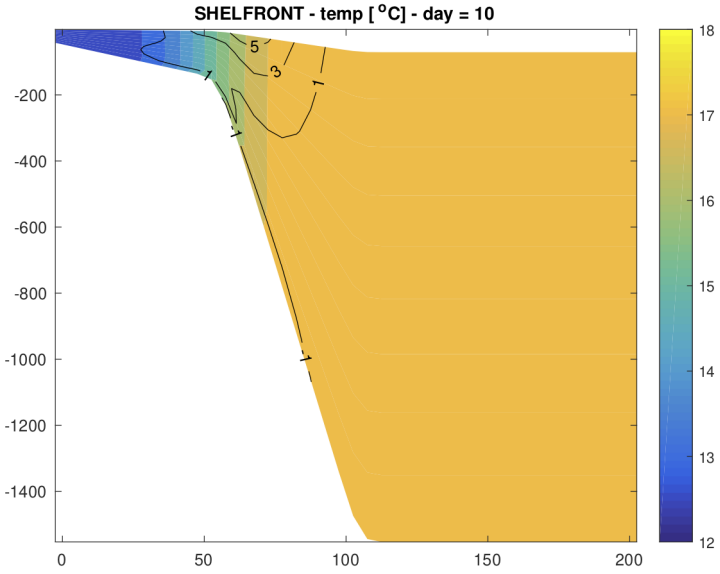


Fig. 9: SHELFRONT results : temerature profile

18.9 Equatorial Rossby Wave

This test problem considers the propagation of a Rossby soliton on an equatorial beta-plane, for which an asymptotic solution exists to the inviscid, nonlinear shallow water equations. In principle, the soliton should propagate westwards at fixed phase speed, without change of shape. Since the uniform propagation and shape preservation of the soliton are achieved through a delicate balance between linear wave dynamics and nonlinearity, this is a good context in which to look for erroneous wave dispersion and/or numerical damping.

The problem is nondimensionalized with: H = 40 cm, L=295 km, T = 1.71 days and U=L/T=1.981 m/s. Theorical propagation speed is 0.4 (0.395) so that at t=120, the soliton should be back to its initial position after crossing the periodic channel of length 48.

Boyd J.P., 1980: Equatorial solitary waves. Part I: Rossby solitons. JPO, 10, 1699-1717

```
# define SOLITON
```

CPP options:

```
# undef  OPENMP
# undef  MPI
# define UV_COR
# define UV_ADV
# define ANA_GRID
# define ANA_INITIAL
# define AVERAGES
# define EW_PERIODIC
# define ANA_SMFLUX
# define NO_FRCFILE
```

Settings :

Results :

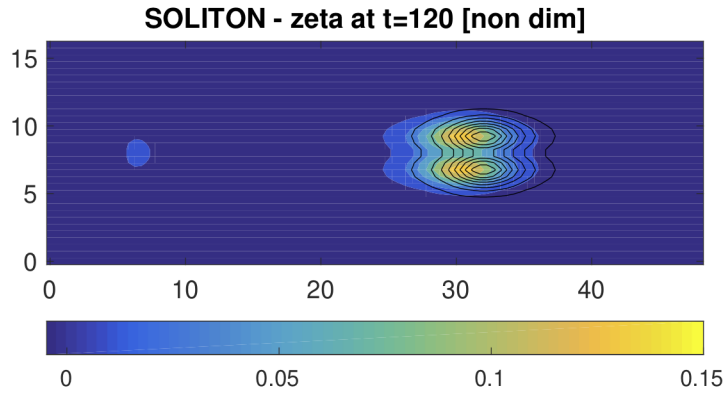


Fig. 10: SOLITON results : sea surface evolution

18.10 Thacker

Thacker, W., (1981), Some exact solutions to the nonlinear shallow-water wave equations. J. Fluid Mech., 107.

```
# define THACKER
```

CPP options:

```
# undef  OPENMP
# undef  MPI
# define THACKER_2DV
# define SOLVE3D
# define UV_COR
# define UV_ADV
# undef  UV_VIS2
# define WET_DRY
# define NEW_S_COORD
# define ANA_GRID
# define ANA_INITIAL
# define ANA_BTFLUX
# define ANA_SMFLUX
# define ANA_SRFLUX
# define ANA_STFLUX
# define NO_FRCFILE
```

Settings :

Results :

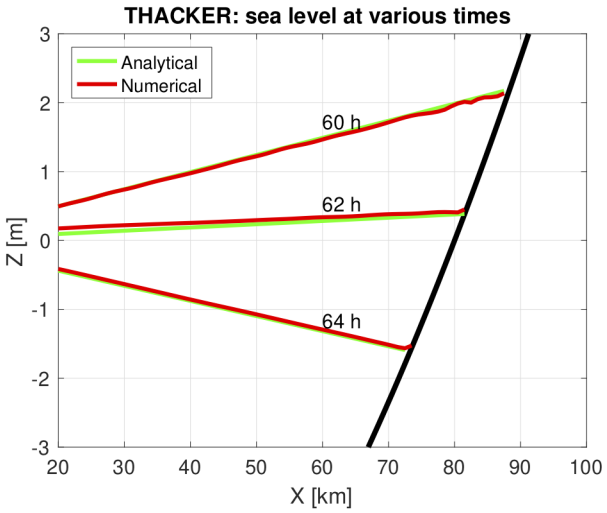


Fig. 11: THACKER results : elevation

18.11 Upwelling

```
# define UPWELLING
```

CPP options:

```
# undef  OPENMP
# undef  MPI
# define SOLVE3D
# define UV_COR
```

(continues on next page)

(continued from previous page)

```
# define UV_ADV
# define ANA_GRID
# define ANA_INITIAL
# define AVERAGES
# define SALINITY
# define NONLIN_EOS
# define ANA_SSFLUX
# define ANA_SRFLUX
# define ANA_STFLUX
# define ANA_BSFLUX
# define ANA_BTFLUX
# define ANA_SMFLUX
# define LMD_MIXING
# define LMD_SKPP
# define LMD_BKPP
# define LMD_RIMIX
# define LMD_CONVEC
# define EW_PERIODIC
# define NO_FRCFILE
```

Settings :

Results :

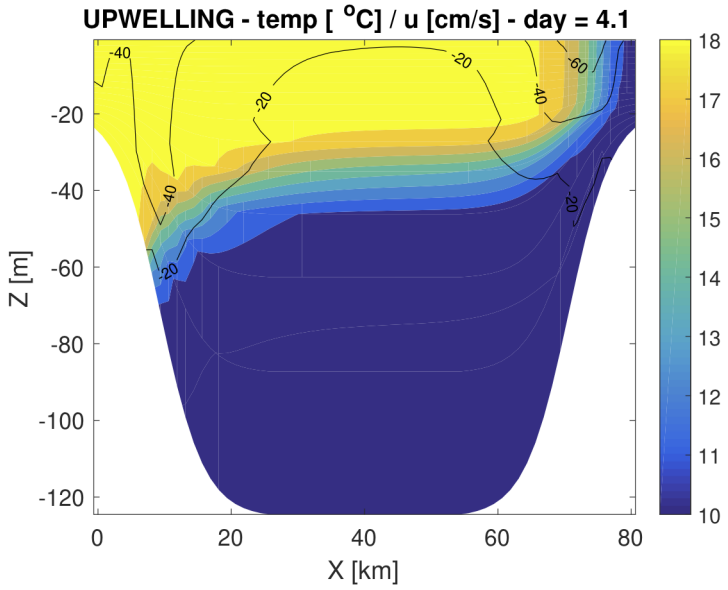


Fig. 12: UPWELLING results : temperature

18.12 Baroclinic Vortex

Free evolution of a baroclinic vortex (South West drift) that retains part of its initial axisymmetric shape as advective effects compensate for weak-amplitude Rossby-wave dispersion in its wake. 1-way and 2-way nesting were tested with this configuration.

McWilliams, J.C., Flierl, G.R., 1979. On the evolution of isolated nonlinear vortices. J. Phys. Oceanogr. 9, 1155–1182.

Penven, P., Debreu, L., Marchesiello, P., McWilliams, J.C., 2006. Application of the ROMS embedding procedure for the central California upwelling system. Ocean Model. 12, 157–187.

Debreu L., P. Marchesiello, P. Penven and G. Cambon, 2012: Two-way nesting in split-explicit ocean models: Algorithms, implementation and validation. Ocean Modelling 49-50 (2012) 1–21

```
# define VORTEX
```

CPP options:

```
# undef  OPENMP
# undef  MPI
# undef  AGRIF
# undef  AGRIF_2WAY
# undef  NBQ
# define SOLVE3D
# define UV_COR
# define UV_ADV
# define ANA_STFLUX
# define ANA_SMFLUX
# define ANA_BSFLUX
# define ANA_BTFLUX
# define ANA_VMIX
# define OBC_EAST
# define OBC_WEST
# define OBC_NORTH
# define OBC_SOUTH
# define SPONGE
# define ZCLIMATOLOGY
# define M2CLIMATOLOGY
# define M3CLIMATOLOGY
# define TCLIMATOLOGY
# define ZNUDGING
# define M2NUDGING
# define M3NUDGING
# define TNUDGING
# define NO_FRCFILE
```

Settings :

Results :

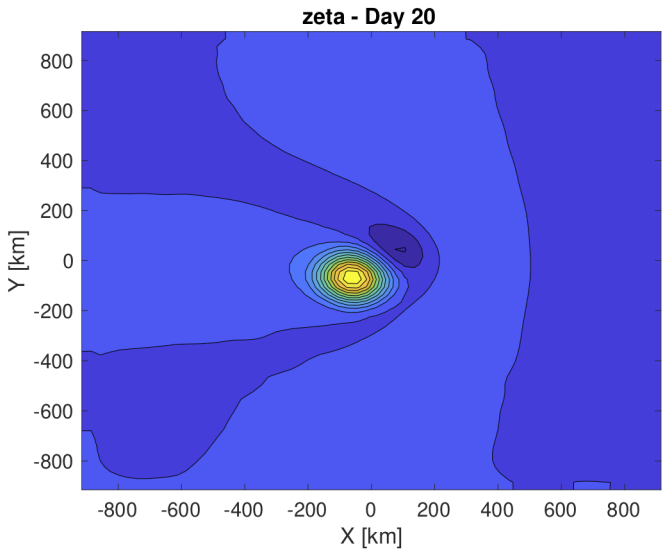


Fig. 13: VORTEX results : difference between parent and child grid (cm)

18.13 Internal Tide

Internal Gravity Wave solution over a ridge.

Di Lorenzo, E, W.R. Young and S.L. Smith, 2006, Numerical and anlytical estimates of M2 tidal conversion at steep oceanic ridges, J. Phys. Oceanogr., 36, 1072-1084.

```
# define INTERNAL
```

CPP options:

```
# undef  OPENMP
# undef  MPI
# define SOLVE3D
# define UV_COR
# define UV_ADV
# define BODYTIDE
# define ANA_GRID
# define ANA_INITIAL
# define ANA_BTFLUX
# define ANA_SMFLUX
# define ANA_SRFLUX
# define ANA_STFLUX
# define ANA_VMIX
# define EW_PERIODIC
# define NS_PERIODIC
# undef  INTERNALSHELF
# ifdef INTERNALSHELF
#  undef   EW_PERIODIC
#  define  OBC_EAST
#  define  OBC_WEST
#  define  SPONGE
#  define  ANA_SSH
#  define  ANA_M2CLIMA
#  define  ANA_M3CLIMA
#  define  ANA_TCLIMA
#  define  ZCLIMATOLOGY
#  define  M2CLIMATOLOGY
#  define  M3CLIMATOLOGY
#  define  TCLIMATOLOGY
#  define  M2NUDGING
#  define  M3NUDGING
#  define  TNUDGING
# endif
# define NO_FRCFILE
```

Settings :

Results :

18.14 Internal Tide (COMODO)

Internal Gravity Wave solution over continental slope and shelf (COMODO test)

Pichon, A., 2007: Tests academiques de maree, Rapport interne n 21 du 19 octobre 2007, Service Hydrographique et Oceanographique de la Marine.

```
# define IGW
```

CPP options:

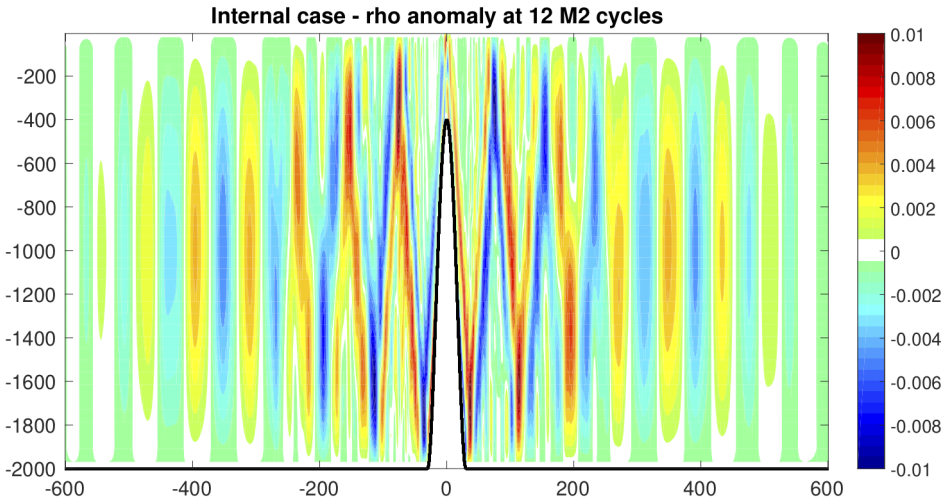


Fig. 14: INTERNAL results : generation of an internal wave

```
# define EXPERIMENT3
# undef  OPENMP
# undef  MPI
# undef  NBQ
# define NEW_S_COORD
# define TIDES
# define TIDERAMP
# define SSH_TIDES
# define UV_TIDES
# define SOLVE3D
# define UV_ADV
# define UV_COR
# define UV_VIS2
# undef  VADV_ADAPT_IMP
# define SPHERICAL
# define CURVGRID
# define ANA_INITIAL
# define ANA_VMIX
# define ANA_SMFLUX
# define ANA_STFLUX
# define ANA_SRFLUX
# define ANA_SSFLUX
# define ANA_BTFLUX
# define ANA_BSFLUX
# define NS_PERIODIC
# define OBC_EAST
# define OBC_WEST
# undef  SPONGE
# define ANA_SSH
# define ANA_M2CLIMA
# define ANA_M3CLIMA
# define ANA_TCLIMA
# define ZCLIMATOLOGY
# define M2CLIMATOLOGY
# define M3CLIMATOLOGY
# define TCLIMATOLOGY
# define M2NUDGING
# define M3NUDGING
# define TNUDGING
# undef  ONLINE_ANALYSIS
```

Settings :

Results :

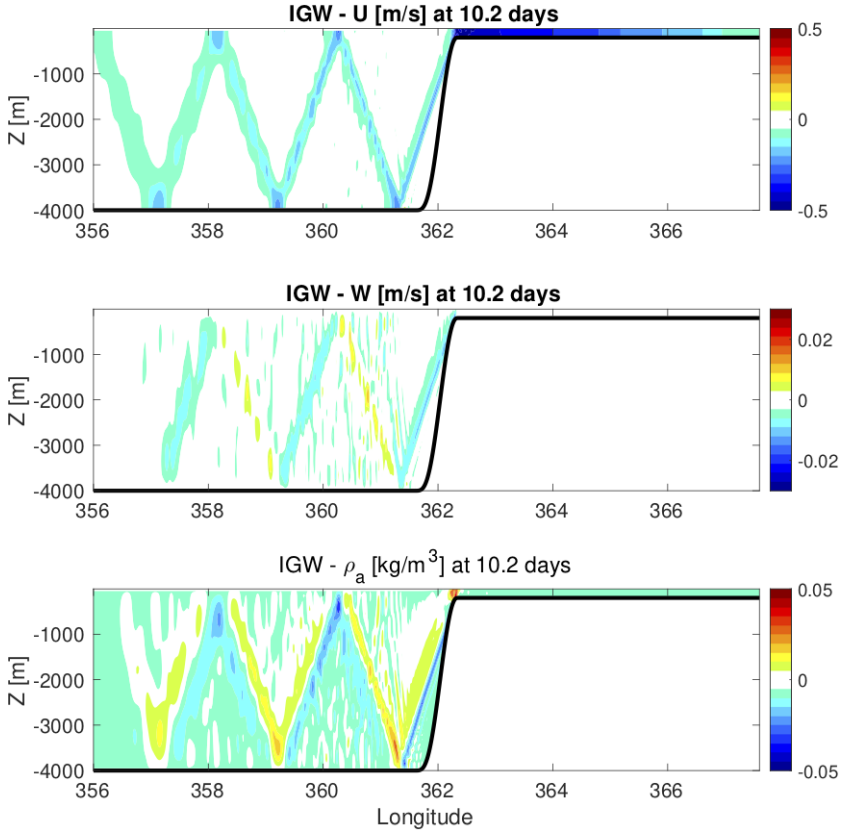


Fig. 15: IGW results : internal gravity waves generation

18.15 Baroclinic Jet

Effective resolution is limited by the numerical dissipation range, which is a function of the model numerical filters (assuming that dispersive numerical modes are efficiently removed). Soufflet et al. (2016) present a Baroclinic jet test case set in a zonally reentrant channel that provides a controllable test of a model capacity at resolving submesoscale dynamics.

A semi-idealized configuration in a periodic channel is set up to generate two dominant mechanisms of upper ocean turbulence: (i) surface density stirring by mesoscale eddies and (ii) fine scale instabilities directly energizing the submesoscale range. The setup consists of a flat reentrant channel of 500 km by 2000 km by 4000 m, centered around 30 deg of latitude on a β-plane (the Coriolis frequency is $1.10^{-4}s^{-1}$ at the center, $\beta = 1.610^{-11}m^{-1}s^{-1}$). Eastern/western boundary conditions are periodic while northern/southern conditions are closed. The initial density field is constructed with interior and surface meridional density gradients and associated geostrophic currents that are linearly unstable to both interior baroclinic and Charney instability modes. A linear stability analysis provides the exponential growth rate of unstable modes as a function of wavenumber. The two most unstable modes are clearly distinct in length scales on either side of the Rossby deformation radius (around 30 km in the center +/- 5 km from south to north). The interior geostrophic instability thus injects energy at mesoscale and Charney instability at submesoscale if resolution allows (2 km). The default resolution is 20 km (40 vertical levels) where only mesoscale instabilities are at work.

Soufflet Y., Marchesiello P., Lemarie F., Jouanno Julien, Capet X., Debreu L., Benshila R. (2016). On effective resolution in ocean models. Ocean Modelling, 98, 36-50.

```
# define JET
```

CPP options:

```
# define ANA_JET
# undef  MPI
# undef  NBQ
# define SOLVE3D
# define UV_COR
# define UV_ADV
# define UV_VIS2
# ifdef ANA_JET
#  define ANA_GRID
#  define ANA_INITIAL
# endif
# define ANA_STFLUX
# define ANA_SMFLUX
# define ANA_BSFLUX
# define ANA_BTFLUX
# define ANA_VMIX
# define EW_PERIODIC
# define CLIMATOLOGY
# ifdef CLIMATOLOGY
#  define ZCLIMATOLOGY
#  define M2CLIMATOLOGY
#  define M3CLIMATOLOGY
#  define TCLIMATOLOGY
#  define ZNUDGING
#  define M2NUDGING
#  define M3NUDGING
#  define TNUDGING
#  define ROBUST_DIAG
#  define ZONAL_NUDGING
#  ifdef ANA_JET
#   define ANA_SSH
#   define ANA_M2CLIMA
#   define ANA_M3CLIMA
#   define ANA_TCLIMA
#  endif
# endif
# define LMD_MIXING
# ifdef  LMD_MIXING
#  undef  ANA_VMIX
#  define ANA_SRFLUX
#  undef  LMD_KPP
#  define LMD_RIMIX
#  define LMD_CONVEC
# endif
# define NO_FRCFILE
```

Settings :

Results :

18.16 Plannar Beach

This test case is a littoral flow driven by obliquely incident waves on a plane beach with a uniform slope of 1:80. The model is forced by monochromatic waves computed with the WKB wave model (Uchiyama et al., 2010) propagating offshore waves with 2 m significant wave height, a peak period of 10 s at an angle of 10° off the shore-normal direction. The horizontal extent of the domain is 1180 m in x (cross- shore), 140 m in y (alongshore) with grid spacings of dx = dy = 20 m. The model coordinates have a west-coast orientation, with the offshore open boundary located at x = 0. The resting depth h varies linearly from 12 m offshore, and is discretized with 20 uniform vertical sigma levels. Boundary conditions are alongshore periodicity, weetin-drying conditions at shore and open boundary conditions at the offshore boundary. Rotation is excluded with f = 0. There is no lateral

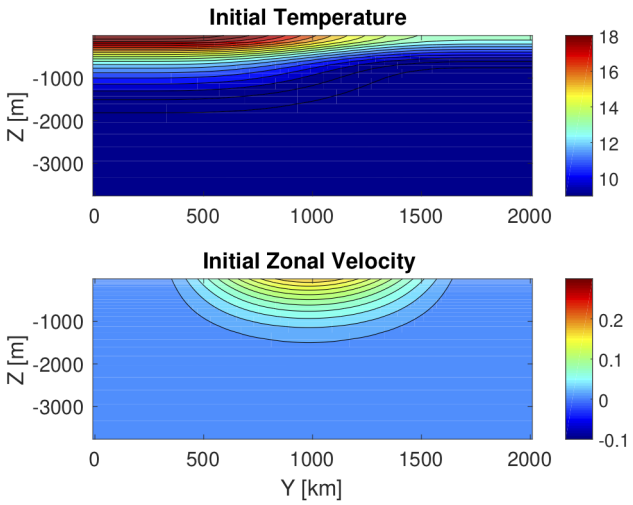


Fig. 16: JET results : initial state

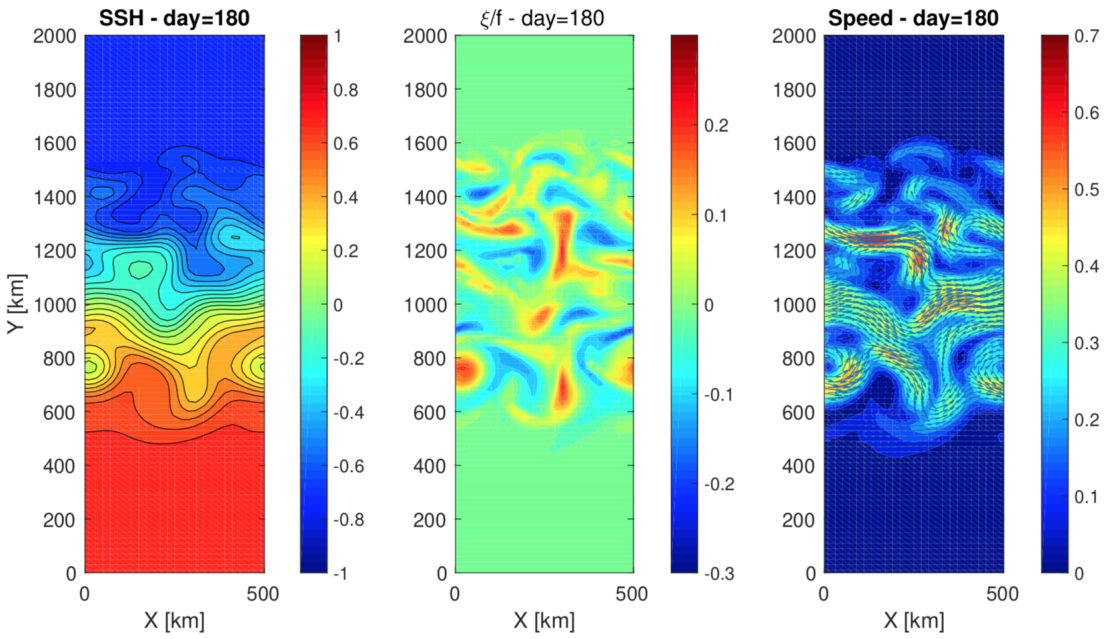


Fig. 17: JET results : results after 180 days

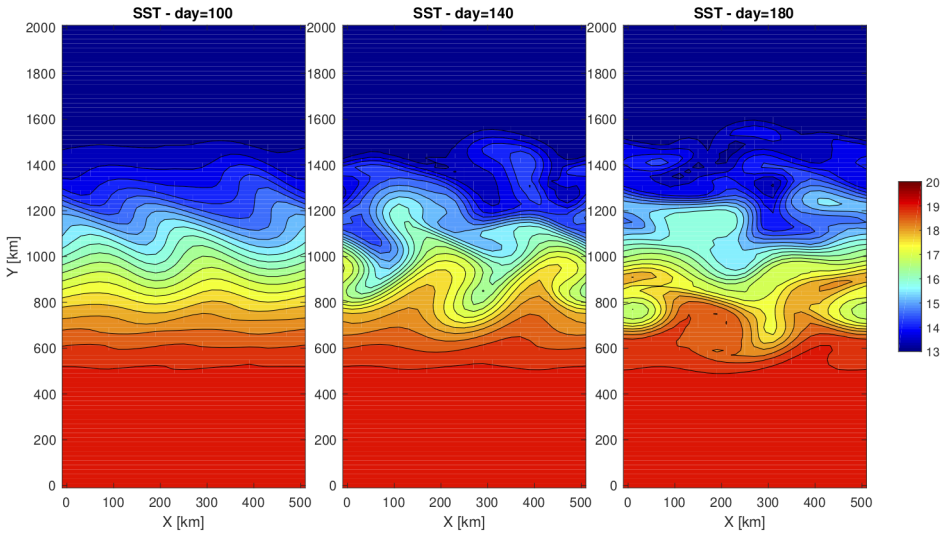


Fig. 18: JET results : vorticity evolutione

momentum diffusion, stratification, nor surface wind/heat/freshwater fluxes. Breaking acceleration is given by the Church and Thornton (1993) formulation in the WKB model and wave-enhanced vertical mixing is computed by the first-order turbulent closure model, K-Profile Parameterization (KPP).

Uchiyama, Y., McWilliams, J.C. and Shchepetkin, A.F. (2010): Wave-current interaction in an oceanic circulation model with a vortex force formalism: Application to the surf zone. Ocean Modelling Vol. 34:1-2, pp.16-35.

```
# define SHOREFACE
```

CPP options:

```
# undef  OPENMP
# undef  MPI
# define SOLVE3D
# define UV_ADV
# undef  MASKING
# define WET_DRY
# define NEW_S_COORD
# define ANA_GRID
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_STFLUX
# define ANA_SSFLUX
# define ANA_SRFLUX
# define ANA_SST
# define ANA_BTFLUX
# define NS_PERIODIC
# define OBC_WEST
# define SPONGE
# define MRL_WCI
# ifdef MRL_WCI
#  undef  WAVE_OFFLINE
#  ifndef WAVE_OFFLINE
#   define WKB_WWAVE
#   define WKB_OBC_WEST
#   define WAVE_FRICTION
#   undef  WAVE_ROLLER
#   undef  MRL_CEW
#  endif
# endif
# define LMD_MIXING
```

(continues on next page)

(continued from previous page)

```
# define LMD_SKPP
# define LMD_BKPP
# undef  BBL
# undef  SEDIMENT
# ifdef SEDIMENT
#  define TCLIMATOLOGY
#  define TNUDGING
#  define ANA_TCLIMA
# endif
```

Settings :

Results :

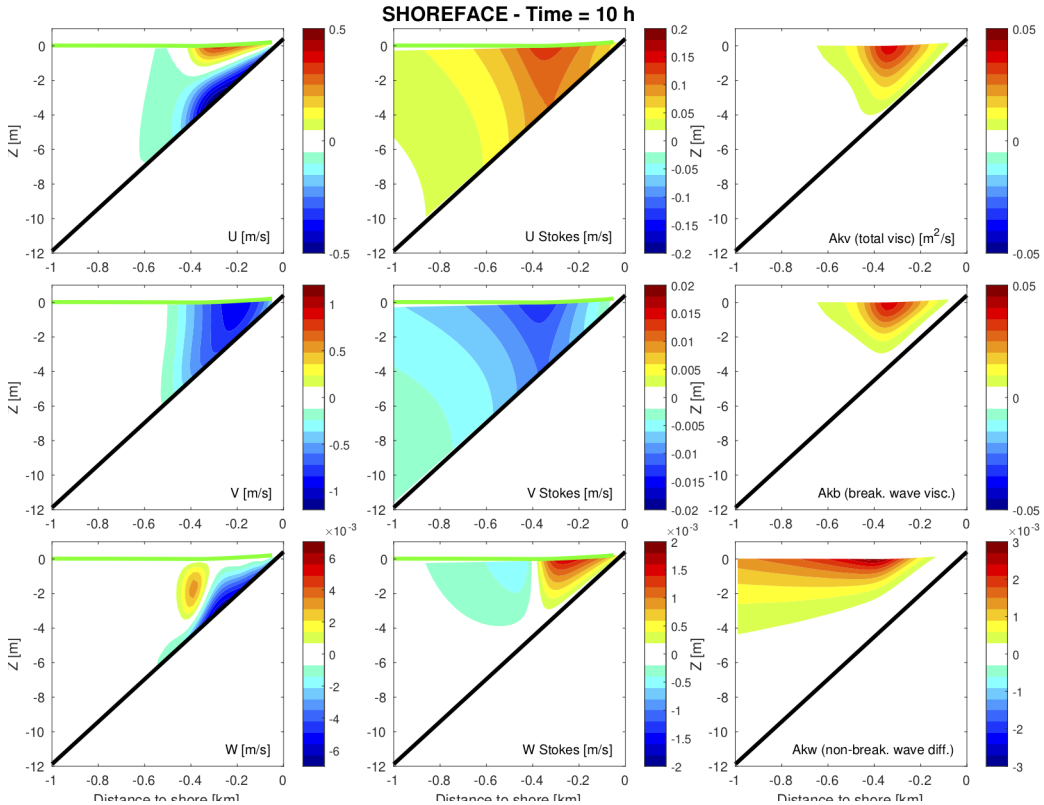


Fig. 19: SHOREFACE results : Eulerian velocities (left), Stokes velocities (center), Vertical mixing (right)

18.17 Rip Current

Rip currents are strong, seaward flows formed by longshore variation of the wave-induced momentum flux. They are responsible for the recirculation of water accumulated on a beach by a weaker and broader shoreward flow. Here, we consider longshore variation of the wave-induced momentum flux due to breaking at barred bottom topography with an imposed longshore perturbation, as in Weir et al. (2010) but in the 3D case. The basin is rectangular (768 m by 768 m) and the topography is constant over time and based on field surveys at Duck, North Carolina. Shore-normal, monochoromatic waves (1m, 10s) are imposed at the offshore boundary and propagate through the WKB wave model coupled with the 3D circulation model (Uchiyama et al., 2011). The domain is periodic in the alongshore direction. We assume that the nearshore boundary is reflectionless, and there is no net outflow at the offshore boundary.

Weir, B., Y. Uchiyama, E. M. Lane, J. M. Restrepo, and J. C. McWilliams (2011), A vortex force analysis of the interaction of rip currents and surface gravity waves, J. Geophys. Res., 116, C05001.

Related CPP options:

RIP	Idealized Duck Beach with 3D topography (default)
BISCA	Semi-realistic Biscarosse Beach (needs input files)
RIP_TOPO_2D	Idealized Duck with longshore uniform topography
GRANDPOPO	Idealized longshore uniform terraced beach (Grand Popo, Benin)
ANA_TIDES	Adds idealized tidal variations
WAVE_MAKER & NBQ	Wave resolving rather than wave-averaged case (#undef MRL_WCI)

CPP options:

```
# define RIP
```

```
# undef  BISCA
# undef  RIP_TOPO_2D
# undef  GRANDPOPO
# ifdef GRANDPOPO
#  define RIP_TOPO_2D
# endif
# undef ANA_TIDES
# undef  OPENMP
# undef  MPI
# define SOLVE3D
# define NEW_S_COORD
# define UV_ADV
# undef  NBQ
# ifdef NBQ
#  define NBQ_PRECISE
#  define WAVE_MAKER
#  define WAVE_MAKER_SPECTRUM
#  define WAVE_MAKER_DSPREAD
#  define UV_HADV_WENO5
#  define UV_VADV_WENO5
#  define W_HADV_WENO5
#  define W_VADV_WENO5
#  define GLS_MIXING_3D
#  undef  ANA_TIDES
#  undef  MRL_WCI
#  define OBC_SPECIFIED_WEST
#  define FRC_BRY
#  define ANA_BRY
#  define Z_FRC_BRY
#  define M2_FRC_BRY
#  define M3_FRC_BRY
#  define T_FRC_BRY
#  define AVERAGES
#  define AVERAGES_K
# else
#  define UV_VIS2
#  define UV_VIS_SMAGO
#  define LMD_MIXING
#  define LMD_SKPP
#  define LMD_BKPP
#  define MRL_WCI
# endif
# define WET_DRY
# ifdef MRL_WCI
#  define WKB_WWAVE
#  define WKB_OBC_WEST
#  define WAVE_ROLLER
#  define WAVE_FRICTION
```

(continues on next page)

(continued from previous page)

```
#  define WAVE_STREAMING
#  define MRL_CEW
#  ifdef RIP_TOPO_2D
#   define WAVE_RAMP
#  endif
# endif
# ifndef BISCA
#  define ANA_GRID
# endif
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_STFLUX
# define ANA_SSFLUX
# define ANA_SRFLUX
# define ANA_SST
# define ANA_BTFLUX
# if !defined BISCA && !defined ANA_TIDES
#  define NS_PERIODIC
# else
#  define OBC_NORTH
#  define OBC_SOUTH
# endif
# define OBC_WEST
# define SPONGE
# ifdef ANA_TIDES
#  define ANA_SSH
#  define ANA_M2CLIMA
#  define ANA_M3CLIMA
#  define ZCLIMATOLOGY
#  define M2CLIMATOLOGY
#  define M3CLIMATOLOGY
#  define M2NUDGING
#  define M3NUDGING
# endif
# ifdef BISCA
#  define BBL
# endif
# undef SEDIMENT
# ifdef SEDIMENT
#  define SUSPLOAD
#  define BEDLOAD
#  undef  MORPHODYN
# endif
# undef  DIAGNOSTICS_UV
```

Settings :

Results :

18.18 Sandbar

This test case is part of an effort to develop a comprehensive 3D nearshore model that predicts onshore and offshore sandbar migrations under storm and post-storm conditions, without the need to modify the model setting parameters. In this test, we attempt to reproduce the results of sandbar migration experiments, the European Large Installation Plan (LIP) experiments, which were carried out at full scale in Delft Hydraulics's Delta Flume (Roelvink and Reniers, 1995). Hydrostatic wave-averaged simulations of LIP-1B (erosion) and LIP-1C (accretion) using CROCO are described in Shafiei et al. (2022), while wave-resolving simulations are in Marchesiello et al. (2021) for hydrodynamics and Marchesiello et al. (2022) for morphodynamics.

In LIP, three types of experiments were carried out under different types of irregular waves, which subsequently

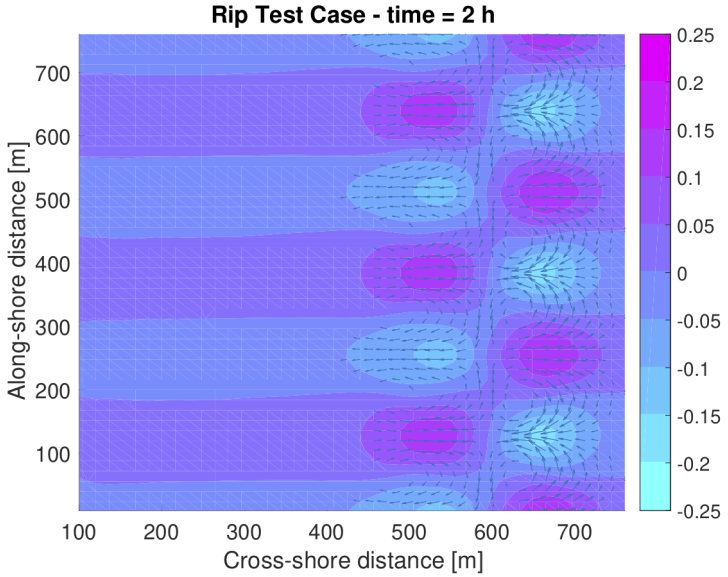


Fig. 20: RIP results : velocity

resulted in a stable (1A), erosive (1B), and accretive (1C) beach state. The initial profile is linear in LIP-1A, with a slope of 1:30 and consisting of a median grain size of 0.22 mm. The final profile of LIP-1A was used as the initial profile of LIP-1B and the final profile of LIP-1B as the initial profile of LIP-1C. The wave conditions were a JONSWAP narrow-banded random wave spectrum with a peak enhancement factor of 3.3 and characteristic wave height and period: Hs =1.4m,Tp =5s (LIP-1B) and Hs =0.6m,Tp =8s (LIP-1C).Under this wave forcing, the sandbar developed during LIP-1B, increasing in height and migrating in the offshore direction. Under the accretive conditions of LIP-1C, the bar migration reversed to the onshore direction. For validation of currents and sand concentration, we consider the time 8 hours after initialization in experiment 1B and 7 hours in 1C. The LIP-1B and LIP-1C experiments lasted 18 and 13 hours, respectively. In both cases, the model was run for one hour with a morphological acceleration factor equal to 18 and 13 respectively.

The model can be run using wave-averaged equations in hydrostatic mode or wave-resolving nonhydrostatic equations.

18.18.1 Wave-averaged solution (default)

Here, wave-averaged equations are used that require parametrization of wave effects on morphodynamics. Bedload nonlinear wave-related transport is parametrized with the SANTOSS formulation, which follows the wave half-cycle concept to account for wave asymmetry and skewness. LIP1b and LIP1c experiments are conducted sequentially, meaning that the final bathymetry of LIP1b is the initial bathymetry of LIP1c. The numerical domain is 200 meters long and 4.1 m deep. The numerical domain is discretized using a uniform grid with horizontal resolution of 1.5 m and the number of vertical layers is 20 throughout the domain (the heights of the cells in the deep region and around the bar are about 21 cm and 5 cm respectively).

For wave forcing, CROCO is fully coupled to built-in ray-theory spectrum-peak wave propagation model. The offshore wave height is forced at the model boundary with values of the experiments. The resulting Dean number $\Omega = H_s/T_p W_s$ clearly differentiates the erosive and accretive conditions. Apart from the forcing conditions, all other wave model parameters are the same for both cases.

For the sediment transport model, the main calibration parameters to be tuned in the suspended load model are: the settling velocity $W_s = 25$ mm/s; the critical bed shear stress $\tau_{CE} = 0.18\,\mathrm{N/m^2}$; and erosion rate $E_0 = 0.001\,\mathrm{kg/m^2/s}$. For bed roughness, the bottom boundary layer model (BBL) uses empirical formulations for sand mobilization based on grain size and wave statistics. For bedload transport, SANTOSS is implemented with only one calibration parameter: the bedload factor, which is set to 0.5.

18.18.2 Wave-resolved solution (#define NBQ)

In this case, we do not rely on parametrization for the bottom boundary layer or bedload transport, as as skewed-asymmetric waves are resolved explicitly, but we make sure that the wave-boundary layer is resolved, and that the first vertical level is in a sheet flow layer (about 10 times the grain size). This is particularly important for the onshore bar migration phase. Note that in our formulation, the turbulence intensity (calculated with the closure model) affects the sediment resuspension. A numerical wave maker forces the JONSWAP spectrum of linear waves at the offshore boundary (as in the laboratory experiments).

Roelvink, J.A., Reniers, 1995. LIP 11D delta flume experiments : a dataset for profile model validation. WL / Delft Hydraulics.

Shafiei H., J. Chauchat, C. Bonamy, and P. Marchesiello, 2022: Numerical simulation of on-shore/off-shore sandbar migration using wave-cycle concept – application to a 3D wave-averaged oceanic model (CROCO), in preparation for Ocean Modelling.

Marchesiello P., J. Chauchat, H. Shafiei, R. Almar, R. Benshila, F. Dumas, 2022: 3D wave-resolving simulation of sandbar migration. Coastal Engineering, submitted.

Marchesiello P., F. Auclair, L. Debreu, J.C. McWilliams, R. Almar, R. Benshila, F. Dumas, 2021: Tridimensional nonhydrostatic transient rip currents in a wave-resolving model. Ocean Modelling, 163, 101816.

```
# define SANDBAR
```

CPP options:

```
# define SANDBAR_OFFSHORE /* LIP-1B */
# undef  SANDBAR_ONSHORE  /* LIP-1C */
# undef  OPENMP
# undef  MPI
# undef  NBQ
# define SOLVE3D
# define UV_ADV
# define NEW_S_COORD
# define ANA_GRID
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_STFLUX
# define ANA_SSFLUX
# define ANA_SRFLUX
# define ANA_SST
# define ANA_BTFLUX
# define OBC_WEST
# define SPONGE
# define WET_DRY
# ifndef NBQ /* ! NBQ */
#  define MRL_WCI
#  ifdef MRL_WCI
#   define WKB_WWAVE
#   define MRL_CEW
#   define WKB_OBC_WEST
#   define WAVE_ROLLER
#   define WAVE_FRICTION
#   define WAVE_BREAK_TG86
#   define WAVE_BREAK_SWASH
#   define WAVE_STREAMING
#   undef  WAVE_RAMP
#  endif
#  define GLS_MIXING
#  define GLS_KOMEGA
#  undef  LMD_MIXING
#  ifdef LMD_MIXING
#   define LMD_SKPP
```

(continues on next page)

(continued from previous page)

```
#   define LMD_BKPP
#   define LMD_VMIX_SWASH
#  endif
#  define BBL
# else /* NBQ */
#  define MPI
#  define NBQ_PRECISE
#  define WAVE_MAKER
#  define UV_ADV
#  define UV_HADV_WENO5
#  define UV_VADV_WENO5
#  define W_HADV_WENO5
#  define W_VADV_WENO5
#  define GLS_MIXING_3D
#  define GLS_KOMEGA
#  define ANA_BRY
#  define Z_FRC_BRY
#  define M2_FRC_BRY
#  define M3_FRC_BRY
#  define T_FRC_BRY
#  define AVERAGES
#  define AVERAGES_K
#  define DIAGNOSTICS_EDDY
# endif /* NBQ */
# define SEDIMENT
# ifdef SEDIMENT
#  define SUSPLOAD
#  define BEDLOAD
#  define MORPHODYN
#  define TCLIMATOLOGY
#  define TNUDGING
#  define ANA_TCLIMA
# endif
# undef  STATIONS
# ifdef STATIONS
#  define ALL_SIGMA
# endif
# undef  DIAGNOSTICS_TS
# ifdef DIAGNOSTICS_TS
#  define DIAGNOSTICS_TS_ADV
# endif
# define NO_FRCFILE
# undef  RVTK_DEBUG
```

Results :

18.19 Swash

This test case addresses wave dynamics on a gently sloping laboratory beach (Globex experiment), using a wave-resolving configuration. The simulation is compared in Marchesiello et al. (2021) against GLOBEX experiments B2 and A3 performed in 2012 in the Scheldt flume of Deltares (Delft, the Netherlands), and described in Michallet et al. (2014). The flume is 110 m long and contains a solid beach of 1:80 slope with its toe at 16.6 m from the wave maker. Experiments are run with a still water depth of 0.85 m and shoreline at x = 84.6 m. Second-order bichromatic waves (B2) are generated at the offshore boundary, with shore normal direction. The grid spacing is dx=1 cm with 10 vertical levels evenly spaced between the free surface and bottom. A simulation with 20 levels gives similar results, while the solution is moderately degraded (mostly in higher moments) with coarser horizontal resolution (dx=3, 6 and 12 cm), which shows good convergence properties. The model time step is dt = 0.15 ms. The minimum depth is 1 mm on the shore, the position of which varies with the swash oscillation, relying on the wetting-drying scheme in CROCO. For bottom drag, the logarithmic law of the wall is used with

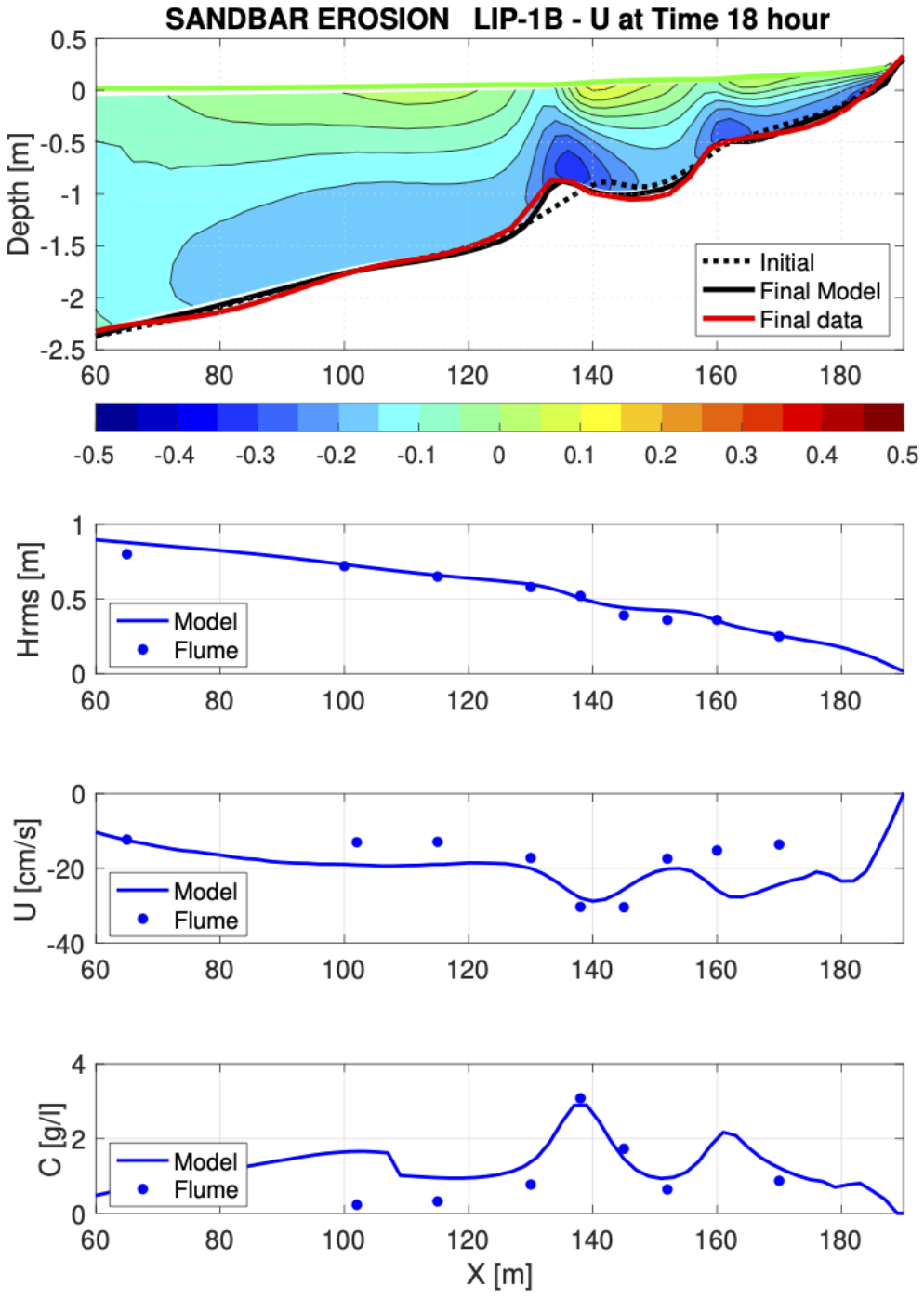


Fig. 21: SANDBAR results : validation of offshore sandbar migration against LIP-1B flume experiment

roughness length $z_0 = 0.0625$ mm.

Marchesiello P., F. Auclair, L. Debreu, J.C. McWilliams, R. Almar, R. Benshila, F. Dumas, 2021: Tridimensional nonhydrostatic transient rip currents in a wave-resolving model. Ocean Modelling, 163, 101816.

Michallet H., B. G. Ruessink, M.V.L.M. da Rocha, A. de Bakker, D. van der A, et al.. GLOBEX: Wave dynamics on a shallow sloping beach. HYDRALAB IV Joint User Meeting, Lisbon, July 2014, Jul 2014, Lisbonne, Portugal.

```
# define SWASH
```

CPP options:

```
# define SWASH_GLOBEX_B2
# undef  SWASH_GLOBEX_A3
# undef  OPENMP
# undef  MPI
# define SOLVE3D
# define AVERAGES
# define NBQ
# define NBQ_PRECISE
# define WAVE_MAKER
# define UV_ADV
# define UV_HADV_WENO5
# define UV_VADV_WENO5
# define W_HADV_WENO5
# define W_VADV_WENO5
# define GLS_MIXING_3D
# define NEW_S_COORD
# define ANA_GRID
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_STFLUX
# define ANA_SSFLUX
# define ANA_SRFLUX
# define ANA_SST
# define ANA_BTFLUX
# define OBC_WEST
# define OBC_SPECIFIED_WEST
# define FRC_BRY
# define ANA_BRY
# define Z_FRC_BRY
# define M2_FRC_BRY
# define M3_FRC_BRY
# define T_FRC_BRY
# define WET_DRY
# define NO_FRCFILE
```

Settings :

Results :

18.20 Tank

The non-hydrostatic solver is tested with several analytical solutions and laboratory experiments. The TANK test case simulates a two-dimensional, deepwater standing wave in a rectangular basin with a depth D and length l of 10 m. The oscillation is caused by a sinusoidal free-surface set-up at time=0. The model uses a uniform grid spacing of 0.2 m in the horizontal and vertical directions. From the dispersion relation ($\sigma = 2\pi/T = gk \tanh kD$, with $L = k/2\pi = 2l$ the wave length), the wave period is T = 3.6 s and phase speed is c = 5.6 m/s. With the hydrostatic assumption, the phase speed and frequency are higher (T = 2.0 s and c = 9.9 m/s). The simulations are compared to analytical solutions.

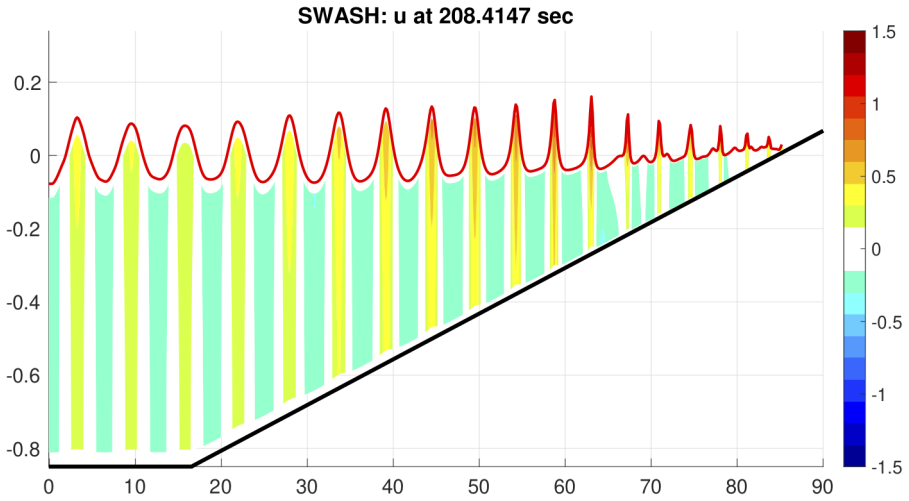


Fig. 22: SWASH results : Velocity and elevation

Chen, X.J., 2003. A fully hydrodynamic model for three-dimensional, free-surface flows. Int. J. Numer. Methods Fluids 42

```
# define TANK
```

CPP options:

```
# undef  MPI
# define NBQ
# ifdef NBQ
#  define NBQ_PRECISE
# endif
# define SOLVE3D
# undef  UV_ADV
# define NEW_S_COORD
# define ANA_GRID
# define ANA_INITIAL
# define ANA_BTFLUX
# define ANA_SMFLUX
# define ANA_SRFLUX
# define ANA_STFLUX
# define NO_FRCFILE
```

Settings :

Results :

18.21 Acoustic wave

```
# define ACOUSTIC
```

CPP options:

```
# undef  MPI
# define NBQ
# ifdef NBQ
#  undef  NBQ_PRECISE
#  define NBQ_PERF
# endif
# undef  UV_VIS2
```

(continues on next page)

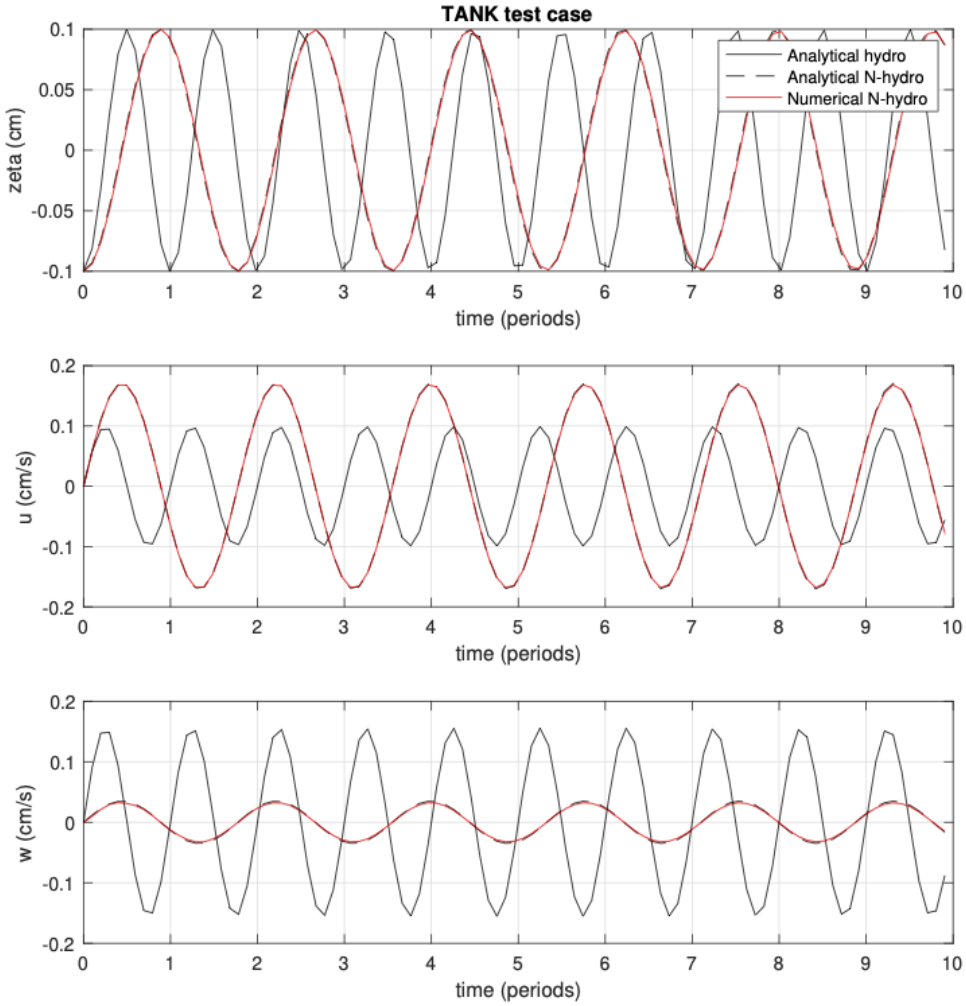


Fig. 23: TANK results : Comparison between analytical, hydrostatique and non-hydrostatique solutions

(continued from previous page)

```
# define SOLVE3D
# define NEW_S_COORD
# define ANA_GRID
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_STFLUX
# define ANA_SRFLUX
# define ANA_BTFLUX
# define NO_FRCFILE
```

18.22 Gravitational Adjustment

The goal of this test case, also known as Lock-Exchange experiment, is to evaluate different numerical advection schemes on representing the adiabatic process in a dam breaking experiment. At the initial time, a vertical density front separates two density classes. Adjustment occurs in which lighter water moves above heavier water (Shin et al., 2004). The model experiments are designed to reproduce the lock-exchange problem described in Ilicak et al., 2012). Analytical solutions to this problem exist and from Bernouilli's equation for an ideal fluid, the front propagates with speed $0.5\sqrt{gH\delta\rho/\rho_0}$. This speed may be slowed down by mixing between the two layers due to numerical diapycnal diffusion.

The setup is a closed, two-dimensional (x,z) domain with a constant depth of H = 20 m and a length of L = 64 km. At t = 0 the two initial densities that represent the two water masses are separated by a vertical barrier. The right and left halves of the domain have densities of 1020 and 1025 kg/m^3 respectively. To investigate the impact of the model resolution and the choice of advection scheme on spurious mixing, the model uses three different horizontal and vertical model grid spacings: coarse (dx=2 km; N=10); medium is default (dx=500 m, N=40); fine (dx=125 m, N=160).

A non-hydrostatique version can be run (#define NBQ) in a smaller domain of 3 m by 30 cm and resolution of 1 cm. In this case, Kelvin-Helmholtz instabilities develop along the front during the gravitational adjustment.

Shin, Dalziel, S., Linden, P, 2004: gravity currents produced by lock exchange. Journal of Fluid Mechanics, 521, 1–34.

Gouillon, F., 2010: Internal Wave Propagation and numerically induced diapycnal mixing in Oceanic general Circulation Models. PhD Thesis at Florida State University, 93pp.

Ilicak, M, Adcroft, A., Griffies, S., Hallberg, R., 2012: Spurious dianeutral mixing and the role of momentum closure. Ocean Modelling, 45–46, 37–58.

```
# define GRAV_ADJ
```

CPP options:

```
# undef  OPENMP
# undef  MPI
# undef  NBQ
# undef  XIOS
# define SOLVE3D
# define NEW_S_COORD
# define UV_ADV
# define TS_HADV_WENO5
# define TS_VADV_WENO5
# define UV_HADV_WENO5
# define UV_VADV_WENO5
# ifdef NBQ
#  define W_HADV_WENO5
#  define W_VADV_WENO5
# endif
# undef  UV_VIS2
```

(continues on next page)

(continued from previous page)

```
# define ANA_GRID
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_STFLUX
# define ANA_BTFLUX
# undef  PASSIVE_TRACER
# define NO_FRCFILE
# undef  RVTK_DEBUG
```

Settings :

Results :

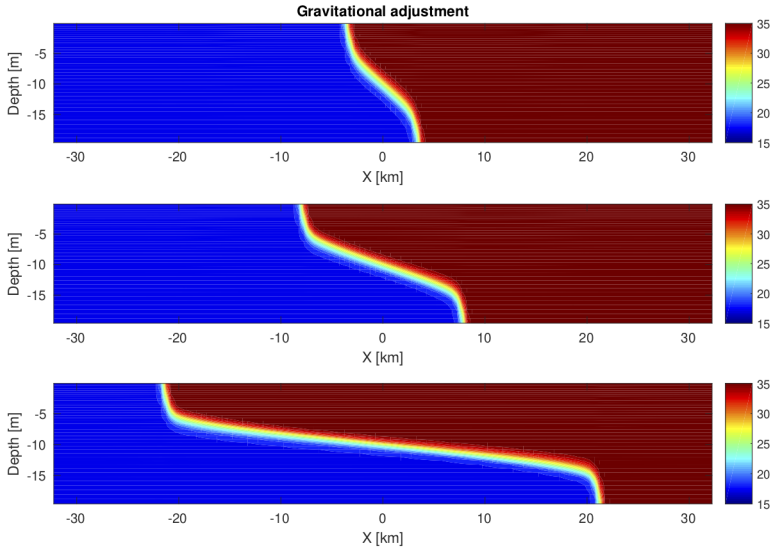


Fig. 24: GRAV_ADJ results : density front evolution for a medium resolution of 500m.

18.23 Internal Soliton

The non-hydrostatic solver is tested with several analytical solutions and laboratory experiments. The Internal Soliton test case is setup from the experiment of Horn et al. (2001). It illustrates the processes acting on an interfacial basin-scale standing wave known as an internal seiche, neglecting the Earth's rotation. The propagation regimes depends on the ratio of the amplitude of the initial wave to the depth of the thermocline, and the ratio of the depth of the thermocline to the overall depth of the basin. In the present setup with moderate wave amplitude, the degeneration mechanism of the basin-scale internal wave is the generation of solitons by nonlinear steepening. As the wave steepens its horizontal lengthscale decreases until the dispersive terms can no longer be neglected. Eventually, a balance between nonlinear steepening and dispersion leads to the evolution of solitary waves, a process described by the Korteweg–de Vries (KdV) equation for the interfacial displacement η_i:

$$\frac{\partial \eta_i}{\partial t} + c_0 \frac{\partial \eta_i}{\partial x} + \alpha \eta_i \frac{\partial \eta_i}{\partial x} + \beta \frac{\partial^3 \eta_i}{\partial x^3}$$

The evolution of solitons is sensitive to the numerical damping associated with the choice of resolution, advection schemes and diffusion operators (implicit in the advection scheme or explicit).

The simulations can be compared with the laboratory experiments of Horn et al. (2001), which were carried out in a tank 6 m long and 29 cm deep. The two-layer fluid is given by a hyperbolic tangent density profile, which is rotated around the center of the basin to initiate the internal seiche at the basin scale. The resolution of the model is 10 cm horizontally and 4 mm vertically.

Horn, D.A., J. Imberger, & G.N. Ivey, (2001). The degeneration of large-scale interfacial gravity waves in lakes. J. Fluid Mech., 434:181-207.

```
# define ISOLITON
```

CPP options:

```
# undef  MPI
# define NBQ
# undef  XIOS
# define SOLVE3D
# define NEW_S_COORD
# define UV_ADV
# define TS_HADV_WENO5
# define TS_VADV_WENO5
# define UV_HADV_WENO5
# define UV_VADV_WENO5
# define W_HADV_WENO5
# define W_VADV_WENO5
# undef  UV_VIS2
# undef  TS_DIF2
# define ANA_GRID
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_STFLUX
# define ANA_BTFLUX
# undef  PASSIVE_TRACER
# define NO_FRCFILE
# undef  RVTK_DEBUG
```

Settings :

Results :

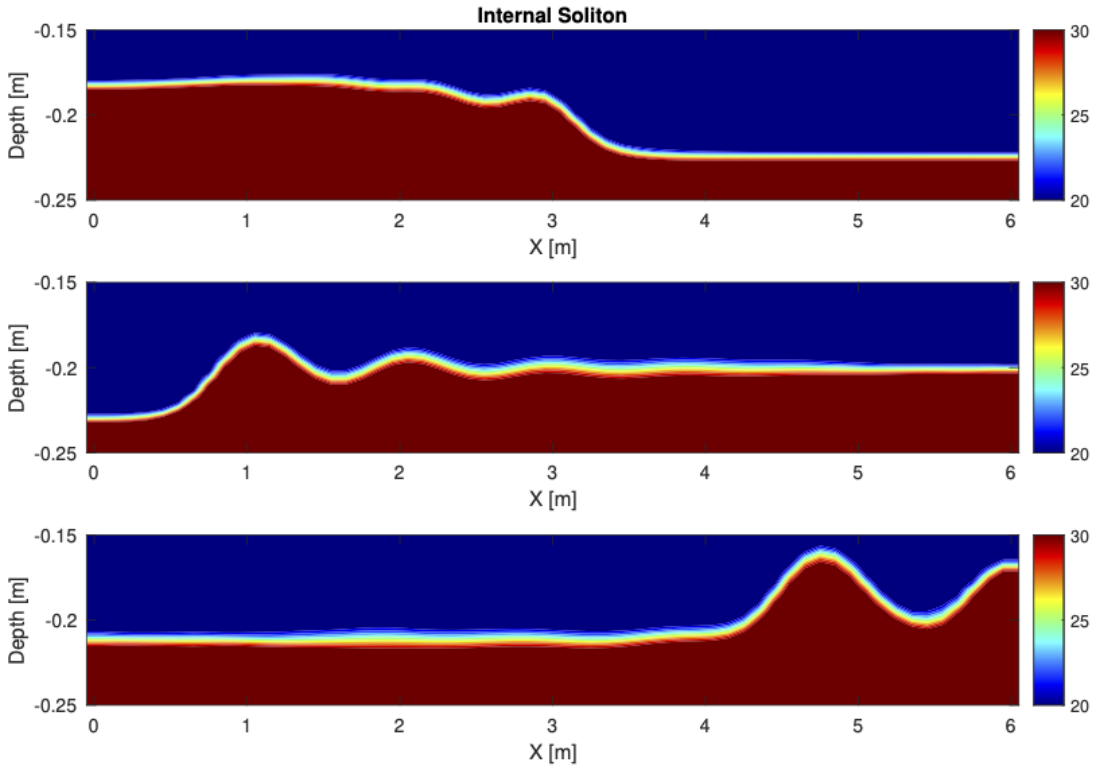


Fig. 25: ISOLITON results : generation of a train of internal solitons from a basin-scale internal seiche

18.24 Kelvin-Helmoltz Instability

This test case runs a Kelvin-Helmholtz instability between two fluid layers. It is part of experiments conducted with CROCO by Penney et al. (2020). The numerical simulations are performed using the non-hydrostatic, non-Boussinesq version of CROCO. While numerical simulations of KH instabilities are often considered in a periodic domain with rigid lid conditionsfor the upper boundary, the implementation presented here uses a free-surface upper boundary, with periodic lateral boundary conditions in the x- and y-directions. The setup is two-dimensional (default) or three-dimensional, with initial density distribution defined as two constant-density layers separated by a strongly stratified pycnocline, with a weak stable background stratification superimposed. The configuration parameters are chosen so that the necessary criterion for stratified shear instability, Ri < 1/4, is satisfied. U(z), the initial background flow providing the shear, is defined by a hyperbolic tangent profile, with the upper layer moving leftward, and the lower layer rightward. Small amplitude perturbations are required to kickstart the instability.

The existence of a free surface and compressibility adds two dynamical processes (surface and acoustic waves) compared to more traditional studies in incompressible, unbounded or rigid lid flows. With the chosen configurations where the instability develops far from the vertical boundaries, the impact of these additional processes is negligible, but in certain circumstances, surface and acoustic waves may play a role in modifying the turbulent cascade.

The results are sensitive to the resolution (1 m by default) and the choice of advection schemes and diffusion operator (implicit in the advection schemes or explicit).

Penney, J., Morel, Y., Haynes, P., Auclair, F., & Nguyen, C. (2020). Diapycnal mixing of passive tracers by Kelvin–Helmholtz instabilities. Journal of Fluid Mechanics, 900, A26.

```
# define KH_INST
```

CPP options:

```
# undef  KH_INSTY
# undef  KH_INST3D
# define MPI
# define NBQ
# define NBQ_PRECISE
# undef  XIOS
# define SOLVE3D
# define NEW_S_COORD
# define UV_ADV
# define TS_HADV_WENO5
# define TS_VADV_WENO5
# define UV_HADV_WENO5
# define UV_VADV_WENO5
# define W_HADV_WENO5
# define W_VADV_WENO5
# undef  SALINITY
# undef  PASSIVE_TRACER
# define ANA_GRID
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_STFLUX
# undef  ANA_SRFLUX
# define ANA_BTFLUX
# define ANA_SSFLUX
# define ANA_BSFLUX
# ifndef KH_INSTY
#  define EW_PERIODIC
# else
#  define NS_PERIODIC
# endif
# define NO_FRCFILE
```

Settings :

Results :

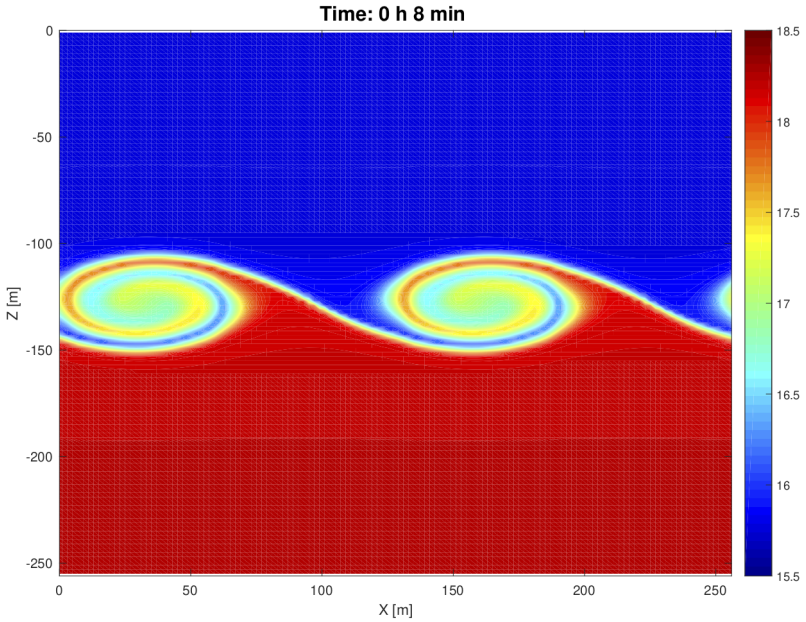


Fig. 26: KH_INST results : instability generation

18.25 Horizontal tracer advection

Test CROCO horizontal advection schemes for tracers

SOLID_BODY_ROT Example with spatially varying velocity DIAGONAL_ADV Constant advection in the diagonal SOLID_BODY_PER Example with a space and time-varying velocity

```
# define TS_HADV_TEST
```

CPP options:

```
# undef  SOLID_BODY_ROT
# undef  DIAGONAL_ADV
# define SOLID_BODY_PER

# undef  OPENMP
# undef  MPI
# undef  UV_ADV
# define NEW_S_COORD
# undef  UV_COR
# define SOLVE3D
# define M2FILTER_NONE
# define ANA_VMIX
# define ANA_GRID
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_SRFLUX
# define ANA_STFLUX
# define ANA_BTFLUX
# define ANA_BSFLUX
# define ANA_SSFLUX
# define NO_FRCFILE
```

(continues on next page)

(continued from previous page)

```
# define SALINITY
# define EW_PERIODIC
# define NS_PERIODIC

# define TS_HADV_UP3
# undef  TS_HADV_C4
# undef  TS_HADV_UP5
# undef  TS_HADV_WENO5
# undef  TS_HADV_C6
```

18.26 Sediment test cases

All the test cases can be defined either with MUSTANG or the USGS sediment model.

---

1. **DUNE cases :**

Migration of a dune composed by different sand classes. Bedload process only

```
# define DUNE
```

CPP options:

```
# undef  OPENMP
# undef  MPI
# define M2FILTER_NONE
# define UV_ADV
# define NEW_S_COORD
# undef  UV_COR
# define SOLVE3D
# define ANA_GRID
# define ANA_INITIAL
# define ANA_SSFLUX
# define ANA_SRFLUX
# define ANA_STFLUX
# define ANA_BSFLUX
# define ANA_BTFLUX
# define ANA_SMFLUX
# define OBC_WEST
# define OBC_EAST
# define ANA_SSH
# define ZCLIMATOLOGY
# define ANA_M2CLIMA
# define M2CLIMATOLOGY
# define GLS_MIXING
# define MORPHODYN
```

=> For Mustang model, just add:

```
# define MUSTANG
# ifdef MUSTANG
#  define key_MUSTANG_V2
#  define key_MUSTANG_bedload
#  define key_tenfon_upwind
# endif
```

=> For USGS sediment model, just add:

```
# define SEDIMENT
# ifdef SEDIMENT
#  undef  SUSPLOAD
#  define BEDLOAD
#  undef  BEDLOAD_WENO5
#  define BEDLOAD_WULIN
#  define TAU_CRIT_WULIN
# endif
```

DUNE case (default):

Dune 2m. Sediment composed of two sand fractions. stratigraphy diagnostics

CPP options to add:

```
# undef  ANA_DUNE      /* Analytical test case (Marieu) */
# undef  DUNE3D        /* 3D example */
```

Results :

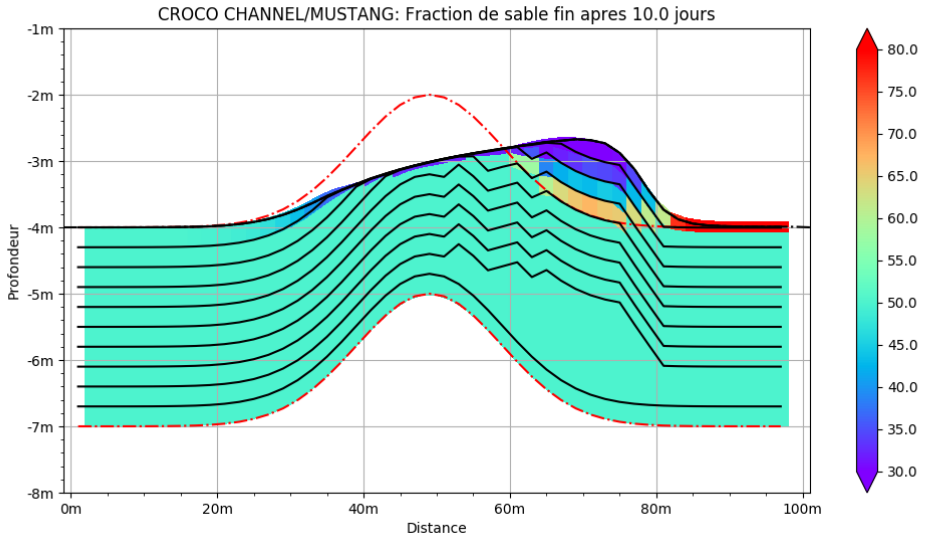


Fig. 27: Fine sand fraction after 10 days in the seabed. The read line indicates the initial position of the dune.

DUNE3D case:

Extension of the DUNE case in 3D. Migration of a Sand bump forced by a barotropic constant flow

CPP options to add:

```
# undef  ANA_DUNE      /* Analytical test case (Marieu) */
# define  DUNE3D       /* 3D example */
```

Results :

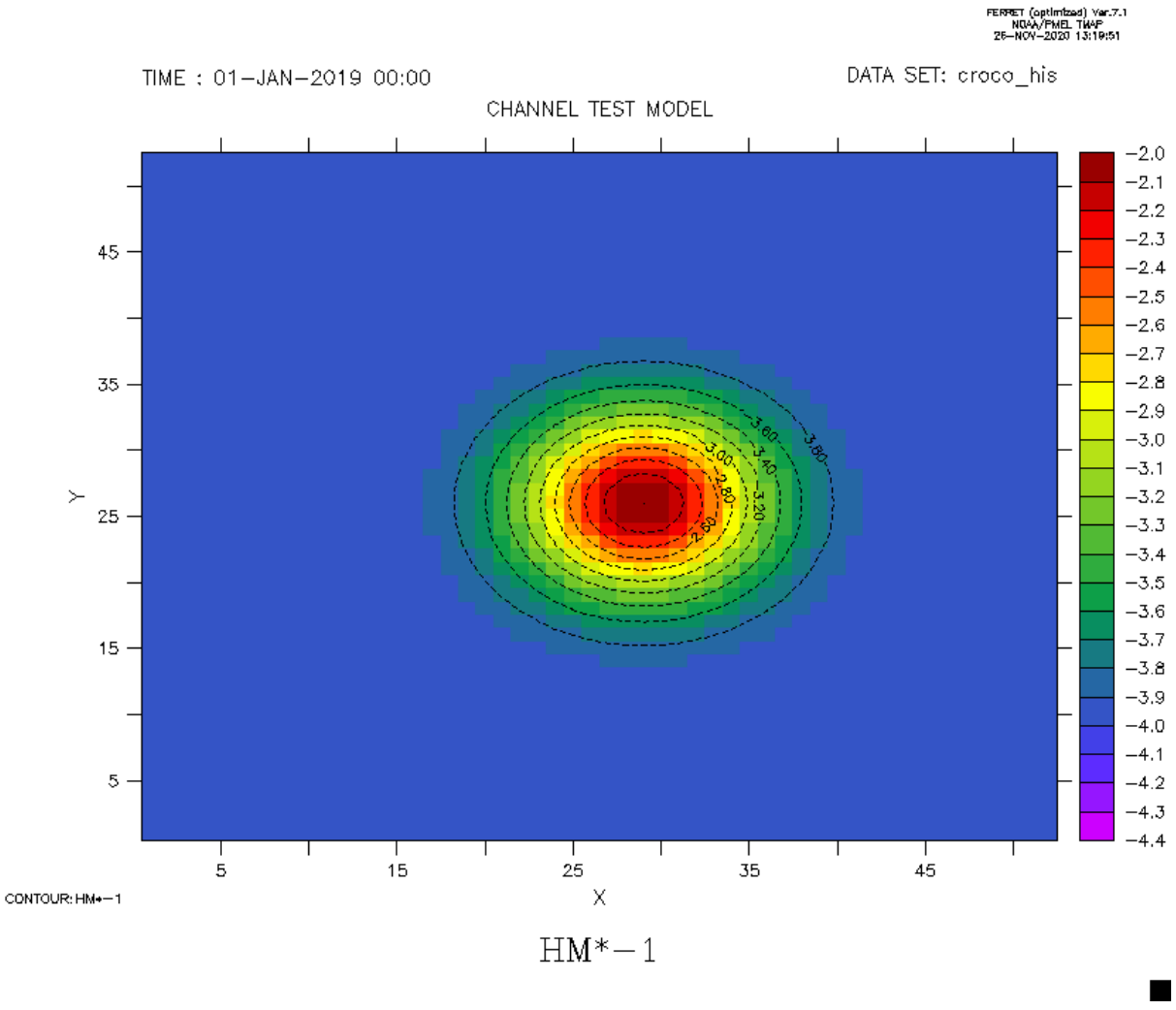


Fig. 28: Sand bump at initialization

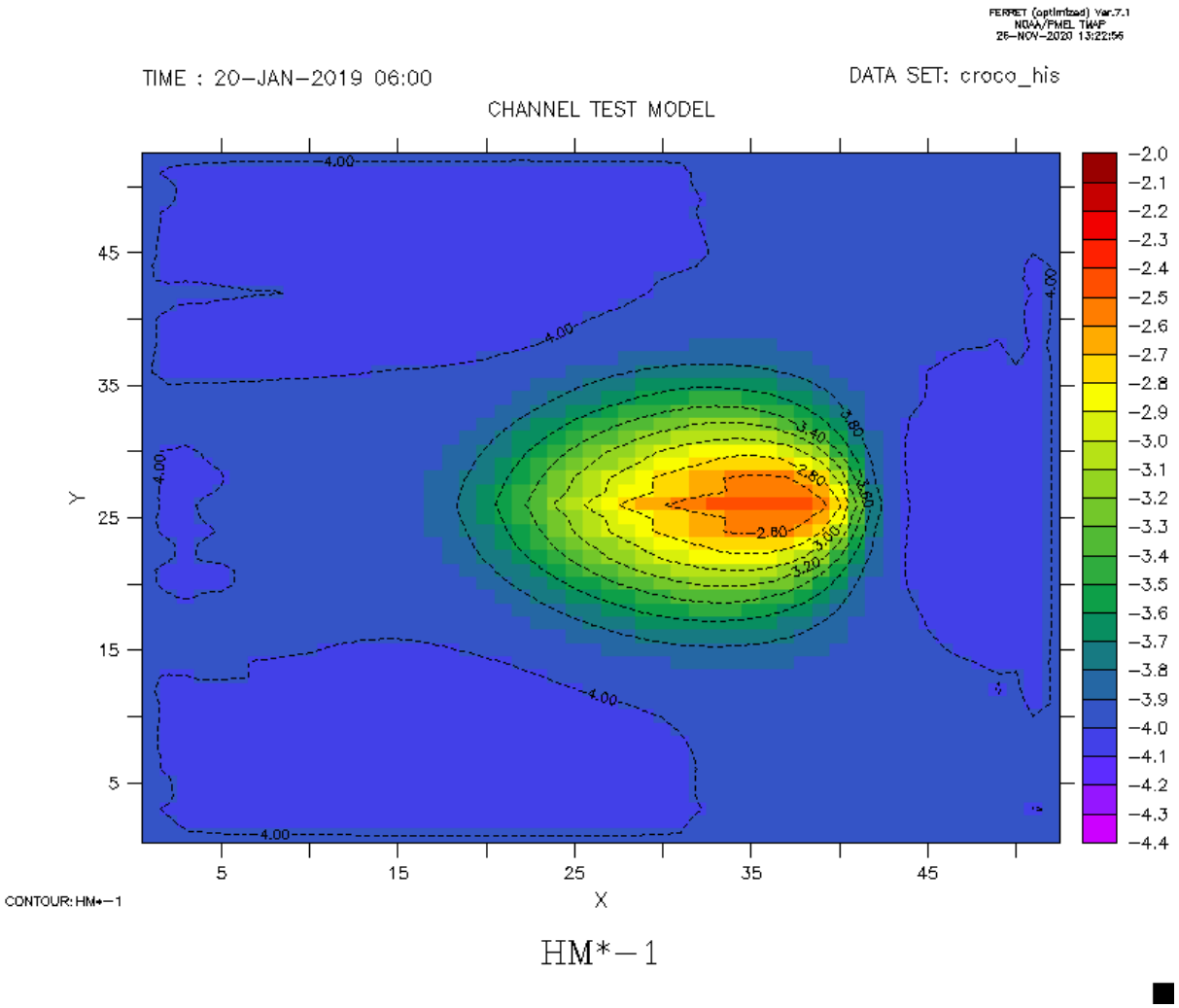


Fig. 29: Sand bump after 20 days

ANA_DUNE case :

Adaptation of the DUNE case. Migration of a sand dune with an analytical bedload formulation that provides an analytical solution for the dune evolution (Long et al., 2008).

Wen Long, James T. Kirby, Zhiyu Shao, A numerical scheme for morphological bed level calculations, Coastal Engineering, Volume 55, Issue 2, 2008, Pages 167-180, https://doi.org/10.1016/j.coastaleng.2007.09.009

CPP options to add:

```
# define ANA_DUNE     /* Analytical test case (Marieu) */
# undef  DUNE3D       /* 3D example */
```

=> For Mustang model, just add:

```
# define MUSTANG
# ifdef MUSTANG
#  define key_MUSTANG_V2
#  define key_MUSTANG_bedload
#  define key_tenfon_upwind
# endif
```

=> For USGS sediment model, just add:

```
# define SEDIMENT
# ifdef SEDIMENT
#  undef  SUSPLOAD
#  define BEDLOAD
#  undef  BEDLOAD_WENO5
#  define BEDLOAD_MARIEU
# endif
```

Results :

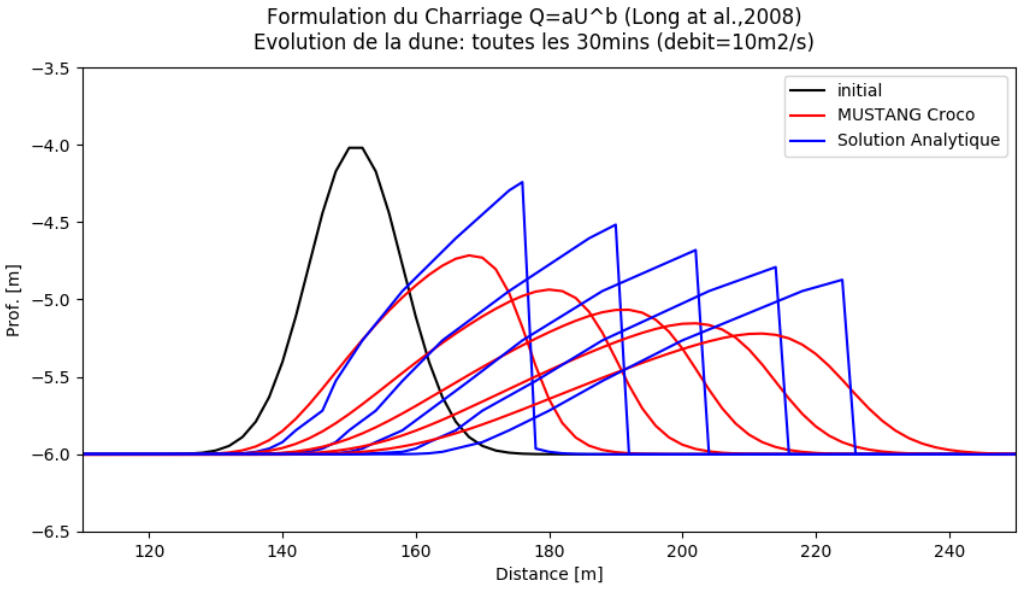


Fig. 30: Comparison between dune propagation (every 30 mins) simulated with CROCO/MUSTANG and computed using the analytical solution

2. **SED_TOY cases :**

Single column test case

```
# define SED_TOY
```

CPP options:

```
# undef  OPENMP
# undef  MPI
# define NEW_S_COORD
# define SOLVE3D
# undef  NONLIN_EOS
# define SALINITY
# undef  UV_VIS2
# define ANA_GRID
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_SRFLUX
# define ANA_STFLUX
# define ANA_SSFLUX
# define ANA_BTFLUX
# define ANA_BSFLUX
# define EW_PERIODIC
# define NS_PERIODIC
```

SED_TOY/ROUSE case :

Testing sediment suspension in a 1DV framework to verify the agreement with Rouse theory

CPP options to add:

```
# define SED_TOY_ROUSE

# define ANA_VMIX
# define BODYFORCE
```

=> For Mustang model, just add:

```
# define MUSTANG
```

=> For USGS sediment model, just add:

```
# define SEDIMENT
# define SUSPLOAD
# define SED_TAU_CD_CONST
```

Results :

SED_TOY/CONSOLID case :

This 1DV test case exemplifies the sequence of depth-limited erosion, deposition, and compaction that characterizes the response of mixed and cohesive sediment in the model. From COAWST experiments, Cohesive and mixed sediment in the Regional Ocean Modeling System (ROMS v3.6) implemented in the Coupled Ocean–Atmosphere–Wave–Sediment Transport Modeling System (COAWST r1234) Sherwood et al., 2018, Geosci. Model Dev., 11, 1849–1871, 2018, https://doi.org/10.5194/gmd-11-1849-2018

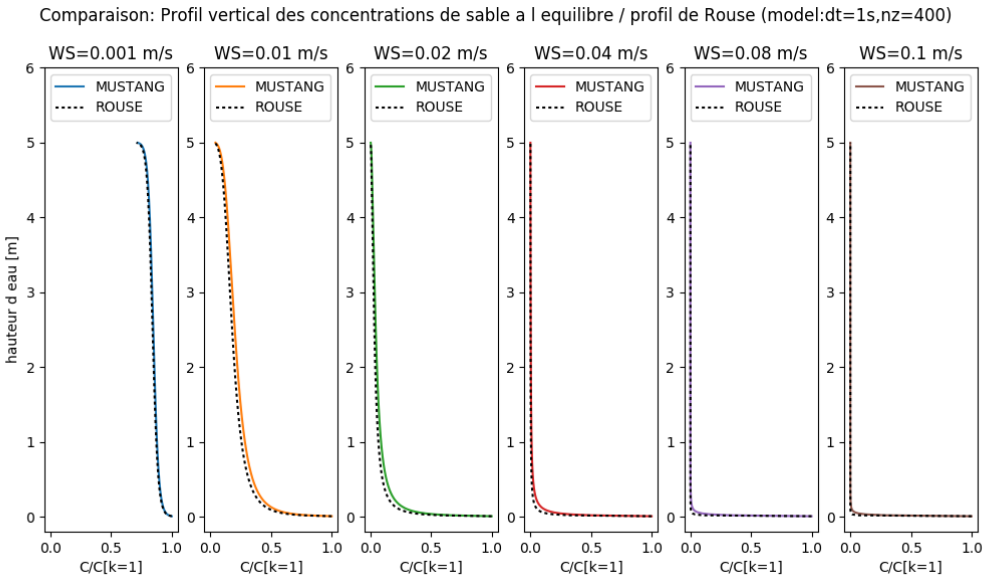


Fig. 31: Comparison between suspended concentration and analytical Rouse profiles for 6 different settling velocities

2 sand classes and 2 mud classes, cohesive behaviour, up to 38 days :

CPP options to add:

```
# define  SED_TOY_CONSOLID

# define SEDIMENT
# define SUSPLOAD
# undef  BBL
# define GLS_MIXING
# define GLS_KOMEGA
# define MIXED_BED
# undef  COHESIVE_BED
```

Results :

SED_TOY/RESUSP case :

This 1DV test case to demonstrate the evolution of stratigraphy caused by resuspension and subsequent settling of different class of sediment during time-dependent bottom shear stress events. From COAWST experiments, Cohesive and mixed sediment in the Regional Ocean Modeling System (ROMS v3.6) implemented in the Coupled Ocean–Atmosphere–Wave–Sediment Transport Modeling System (COAWST r1234) Sherwood et al., 2018, Geosci. Model Dev., 11, 1849–1871, 2018, https://doi.org/10.5194/gmd-11-1849-2018

2 sand classes and 2 mud classes, non cohesive behaviour:

CPP options to add:

```
# define  SED_TOY_RESUSP

# define SEDIMENT
# define SUSPLOAD
# undef  BBL
```

(continues on next page)

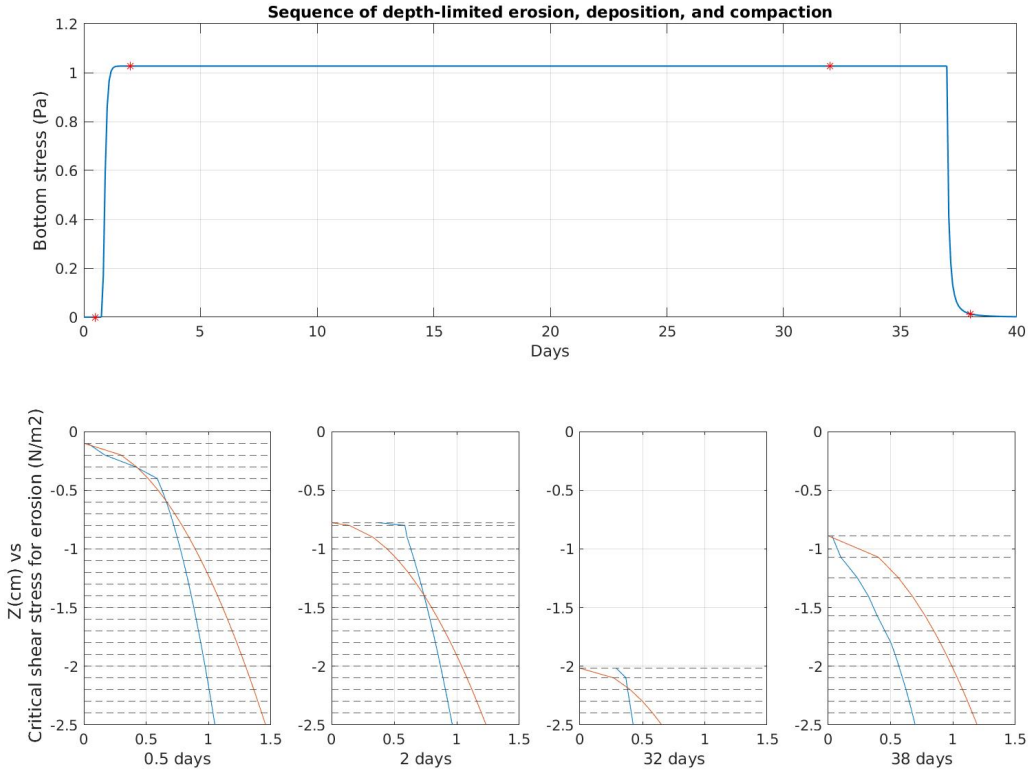


Fig. 32: Evolution of equilibrium bulk critical stress profile for erosion (red solid line) and the instantaneous profile of bulk critical stress for erosion (blue solid line)

(continued from previous page)

```
# define GLS_MIXING
# define GLS_KOMEGA
# define MIXED_BED
# undef  COHESIVE_BED
```

Results :

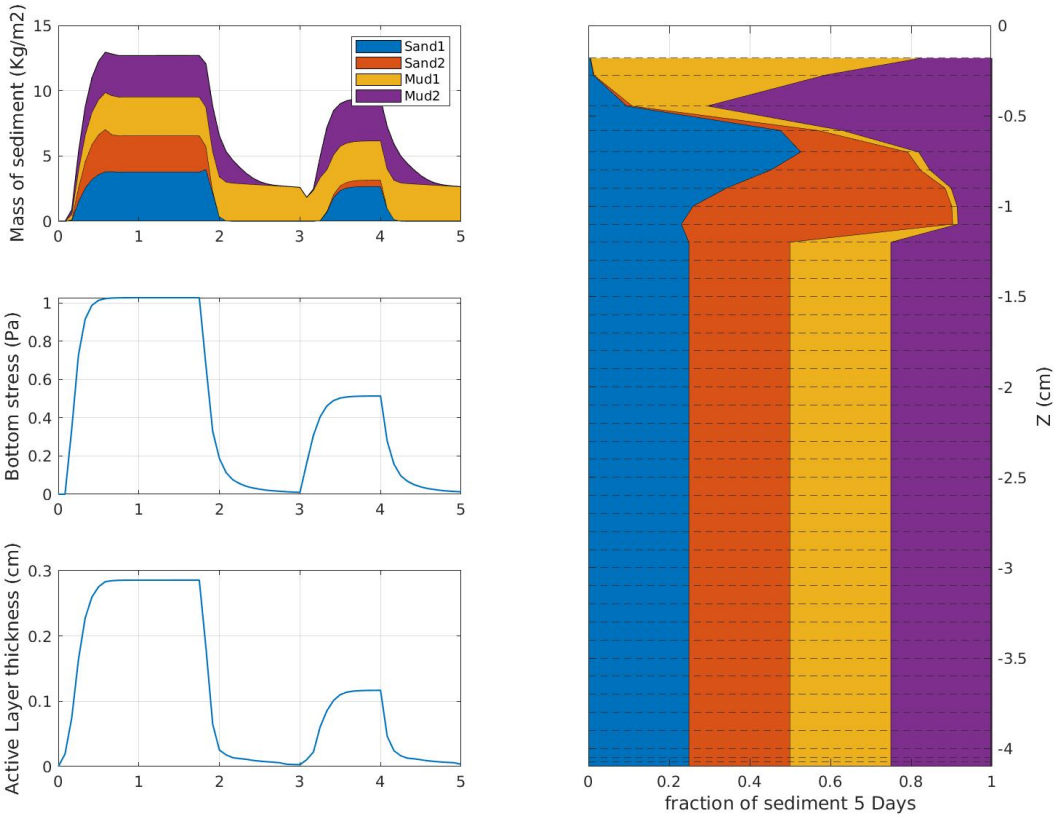


Fig. 33: Double surface stress event and response on stratigraphy 5 days later

3. **TIDAL_FLAT case :**

   2DV tidal flat with a sediment mixture (mud, fine sand, medium sand) - suspension only

```
# define  TIDAL_FLAT
```

   CPP options:

```
# undef  OPENMP
# undef  MPI
# undef  NONLIN_EOS
# define NEW_S_COORD
# define SALINITY
# define UV_ADV
# define TS_HADV_WENO5
# define TS_VADV_WENO5
# define UV_HADV_WENO5
```

(continues on next page)

(continued from previous page)

```
# define UV_VADV_WENO5
# define UV_COR
# define SOLVE3D
# define UV_VIS2
# define GLS_MIXING
# define ANA_INITIAL
# define WET_DRY
# define TS_DIF2
# define SPONGE
# define ANA_GRID
# define ANA_INITIAL
# define ANA_SMFLUX
# define ANA_SRFLUX
# define ANA_STFLUX
# define ANA_SSFLUX
# define ANA_BTFLUX
# define ANA_BSFLUX
# define OBC_WEST
# define FRC_BRY

# define MUSTANG
# ifdef MUSTANG
#  define key_sand2D
#  undef  key_MUSTANG_V2
# endif
```

3. **FLOCMOD cases :**

   FLOCMOD 0D – comparison with laboratory experiments [#SED_TOY_FLOC_0D]

   This test case simulates a laboratory experiment dedicated to flocculation experiments under controlled conditions (see Verney et al., 2011). Natural SPM were mixed in a jar and agitation was tuned to simulate turbulence variations during a tidal cycle. G values ranged from 1 s-1 (around slack periods) to 12 s-1 (during flood/ebb periods). Floc size were monitored using a CCD camera, and PSD were extracted from image processing routines.

   This test case can be activated with the cppkey #SED_TOY_FLOC_0D. This test case has a 1DV structure but current is set to 0 (no advection, no diffusion), and settling is not allowed (Ws = 0 m.s-1 for all classes). Shear rate is imposed using experimental values in each vertical grid cell.

   The initial concentration is set to 0.093 g.l-1 (experimental value) and the initial distribution is spread over the floc size classes lower than 50 μm.

   15 floc classes are used, logarithmically distributed from 4 μm to 1500 μm. Primary particle size is set to 4 μm and nf = 1.9. CROCO time step is set to 2 s.

   Shear aggregation and binary shear fragmentation only

   FLOCMOD main parameters are : alpha = 0.43 and beta = 0.1.

   Initial floc size distribution is far from the equilibrium, and FLOCMOD fails to reproduce the first flocculation period. After the first flood period, FLOCMOD and experimental results are in good agreement considering the D50. The settling phase observed in the experiment from 06:00 to 07:00 is not simulated in the 0D model.

   Shear aggregation, binary shear fragmentation and floc erosion

   FLOCMOD main parameters are : alpha = 0.35 , beta = 0.1 , f_ero_frac = 0.4 and f_ero_iv = 3.

   FLOCMOD reference is the shear aggregation/fragmentation test detailed above.

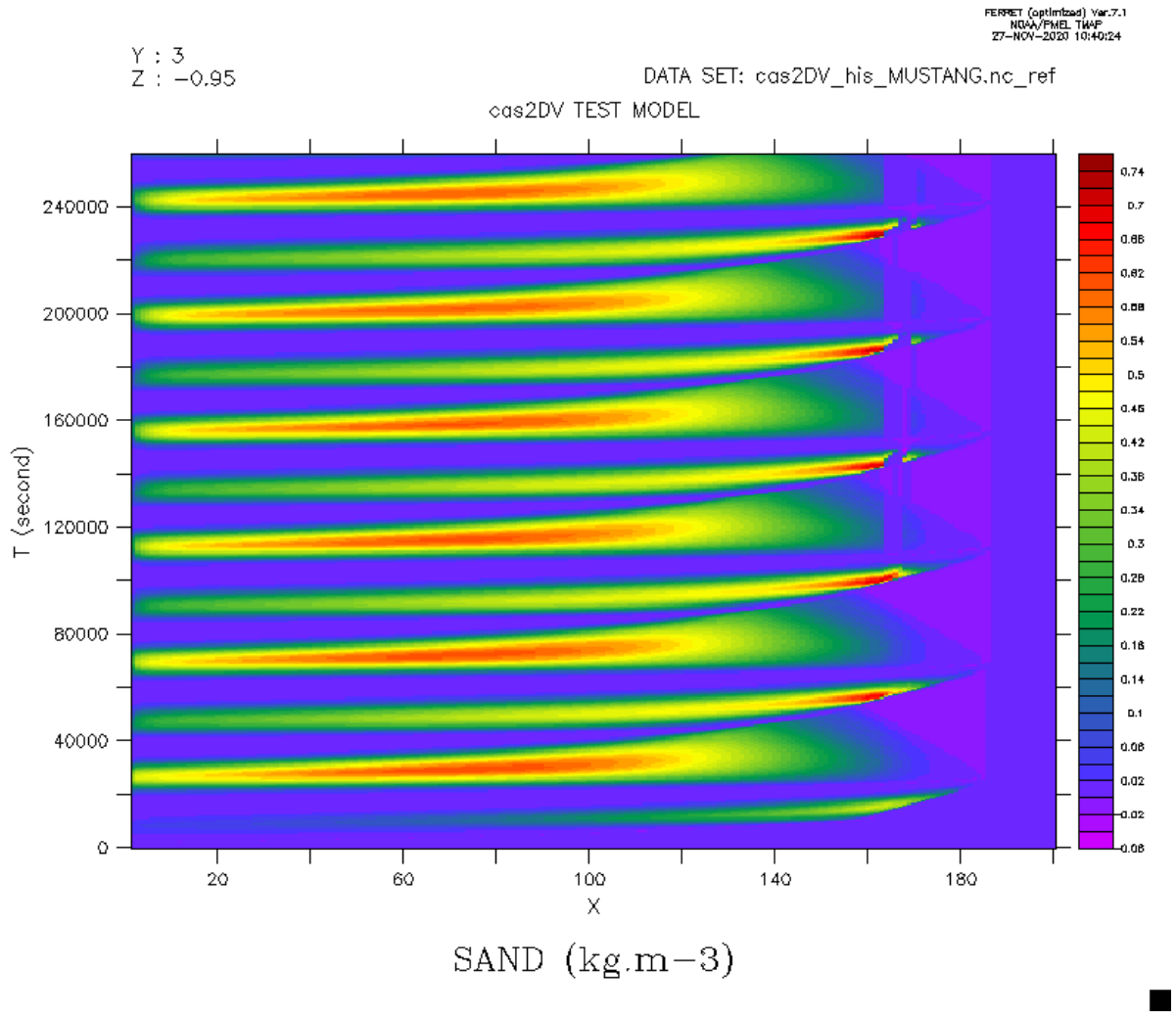
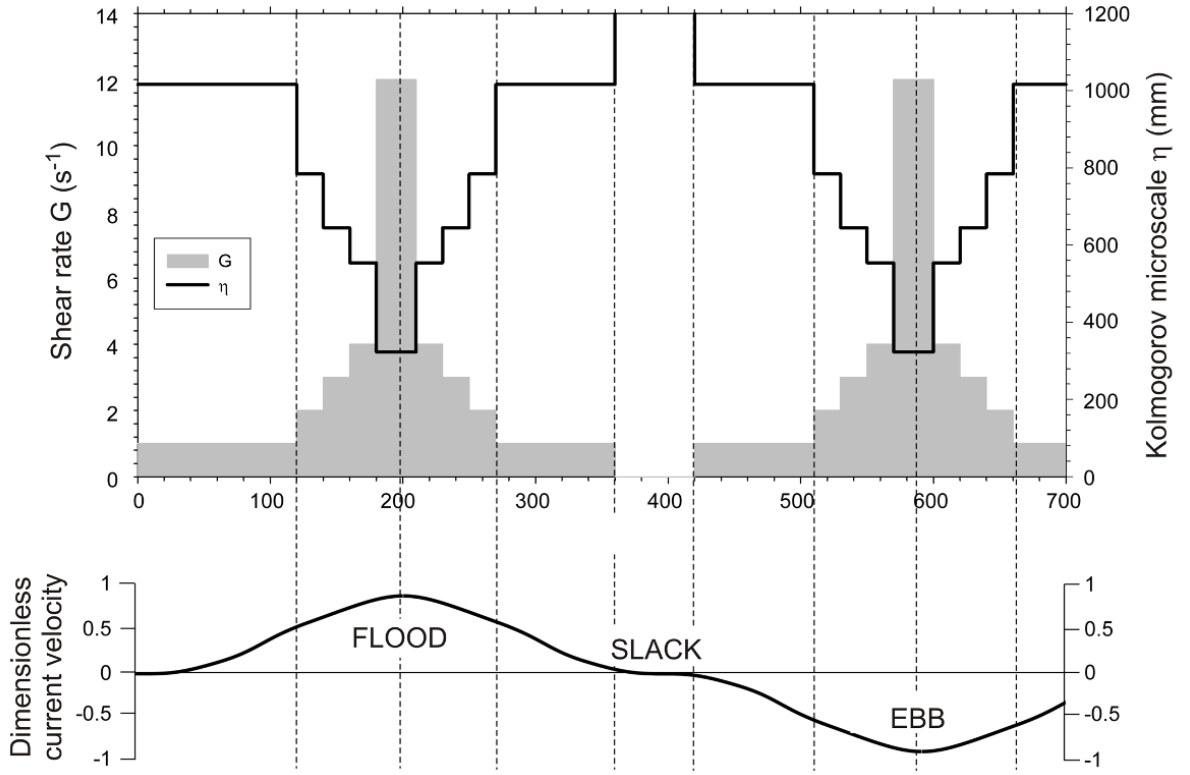
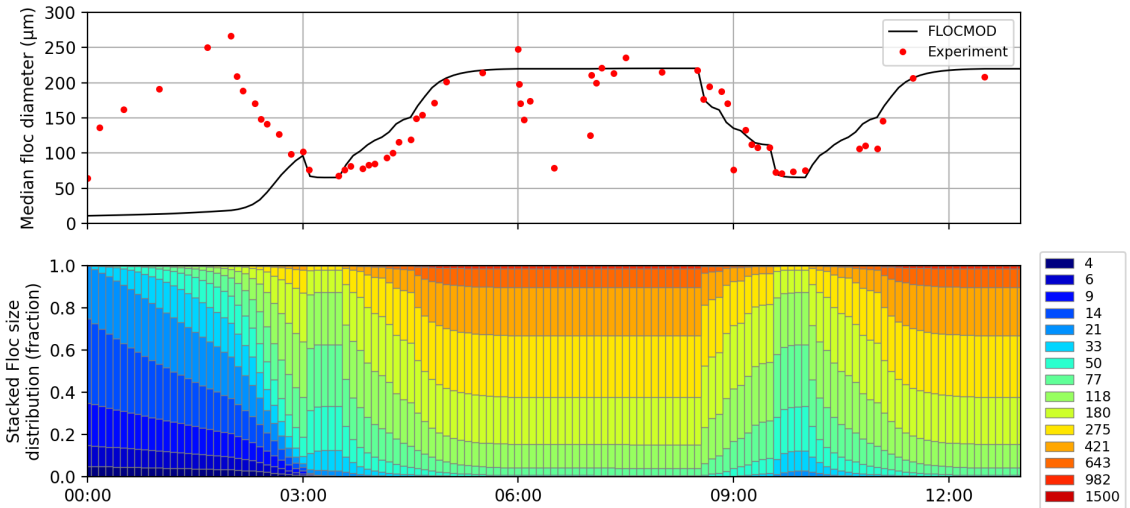
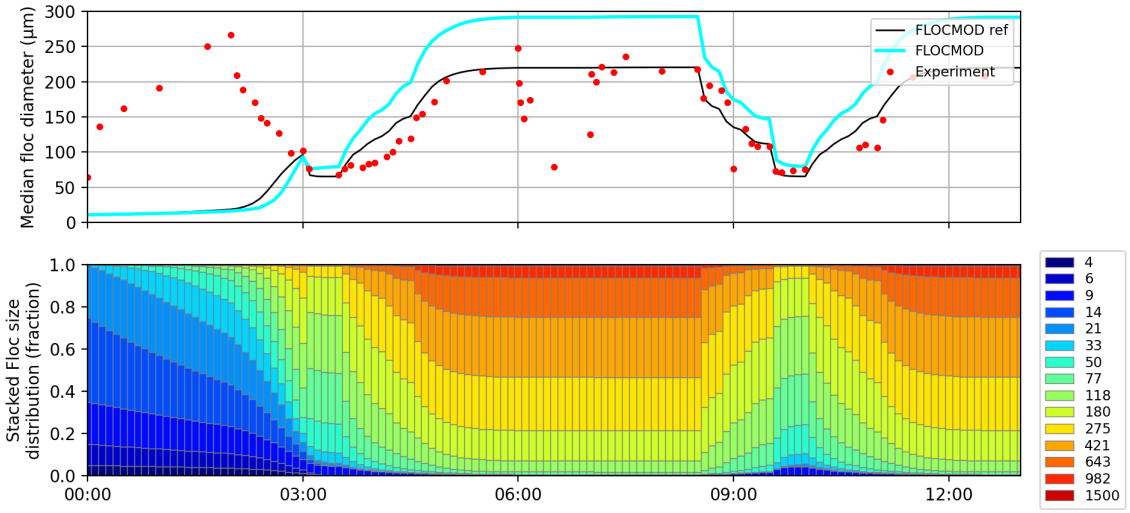


Fig. 34: Bottom mud concentration evolution over several tidal cycles


Shear rate G (s-1)
Kolmogorov microscale η (mm)
G
η
Dimensionless current velocity
FLOOD
SLACK
EBB


Median floc diameter (µm)
FLOCMOD
Experiment
Stacked Floc size distribution (fraction)
4
6
9
14
21
33
50
77
118
180
275
421
643
982
1500



The median floc size dynamics is globally similar to the reference, however flocculation is more intense as part of shear fragmentation is attributed to floc erosion, hence flocs are less fragmented globally. We can also notice that erosion mode maintains a bimodal distribution, with a microfloc population (due to erosion) and macrofloc population varying in time with turbulence.

FLOCMOD 1DV [#SED_TOY_FLOC_1D]

This test case illustrates the vertical flocculation dynamics along a tidal cycle. 15 floc classes are used, logarithmically distributed from 4 μm to 1500 μm. Primary particle size is set to 4 μm and nf = 1.9. CROCO time step is set to 10s.

CROCO main parameters are : h = 5 m, 50 vertical layers. A sinusoidal forcing is applied.

MUSTANG parameters : E0 = 0.0005 kg.m-3 and toce = 0.3N.m-2.

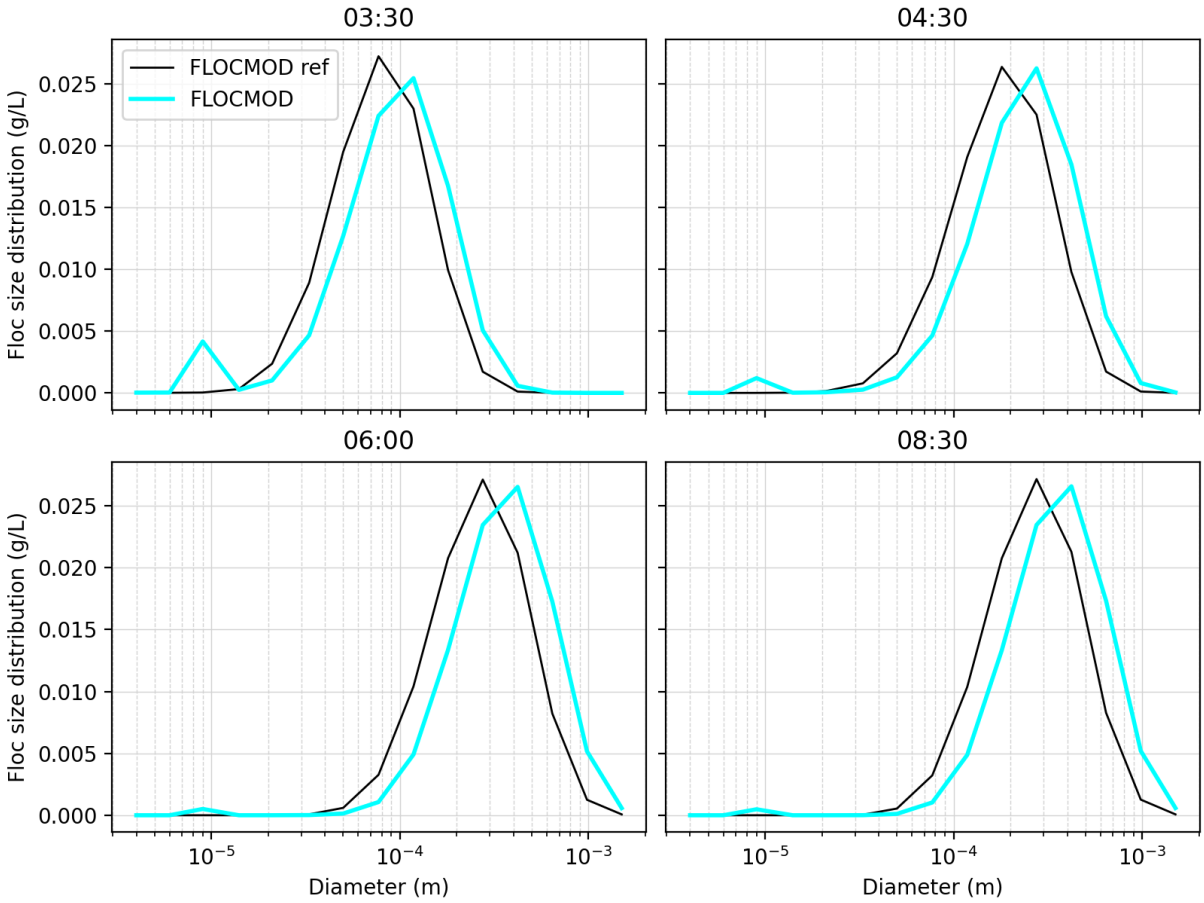
FLOCMOD main parameters are : alpha = 0.4 and beta = 0.2, including shear erosion and binary fragmentation: f_ero_frac = 0.5 and f_ero_iv = 4.

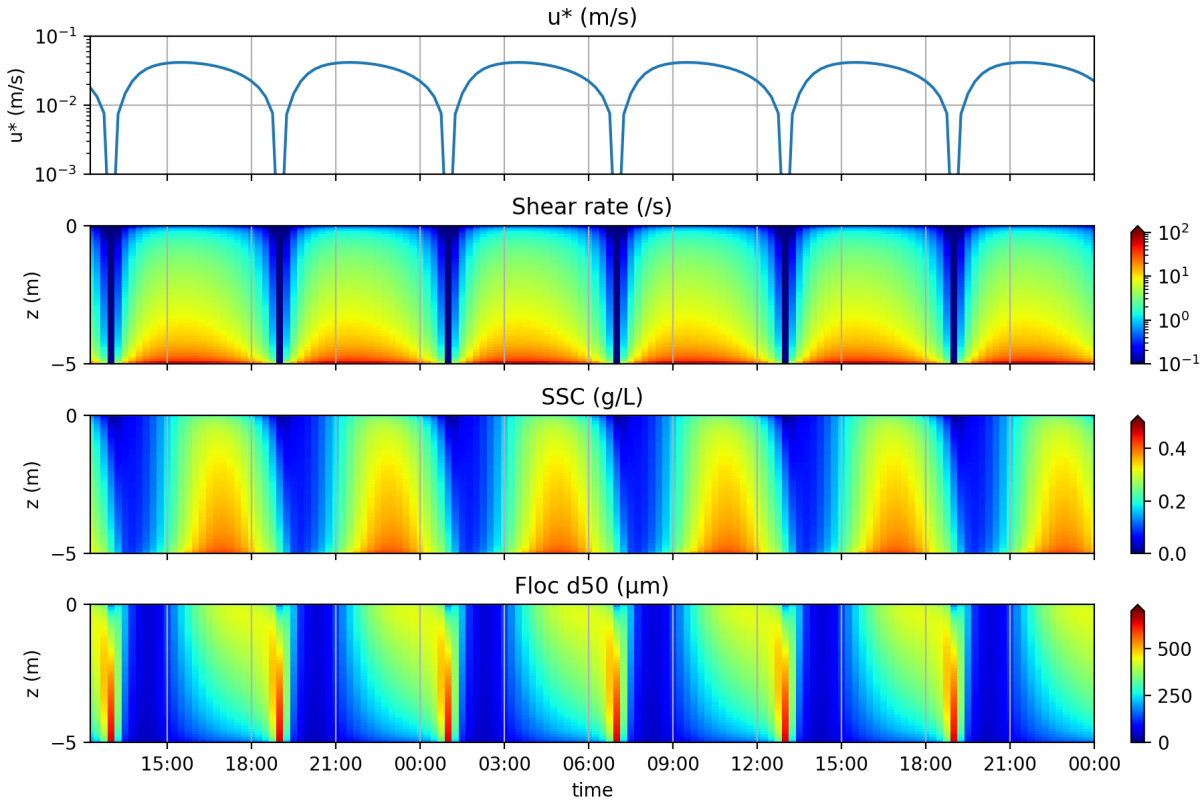
The shear rate varied from O(0.1 s-1) during slack to O(50 s-1) during max flood/ebb currents close to the bed. In the upper part of the water column, the shear rate is lower, and reaches up to O(1 s-1) during maximum current velocities.

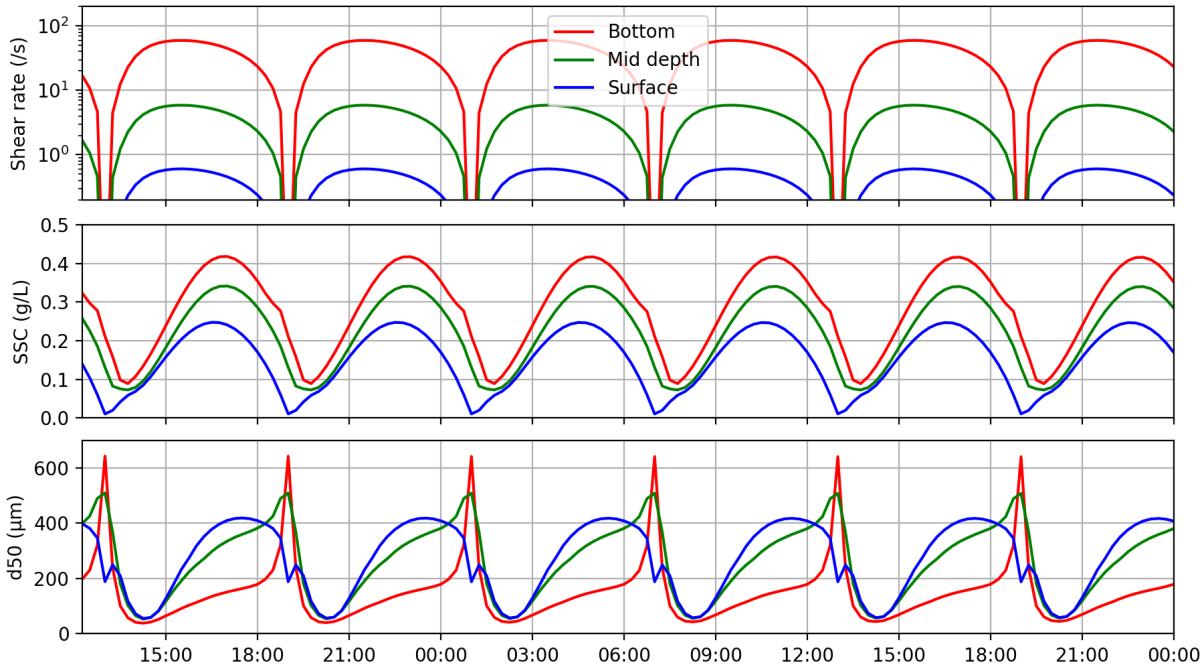
This tidal forcing induces resuspension during high shear stress periods, and SSC reaches up to 0.5 g/L close to the bed. Floc size distribution strongly varies along the tide, with the smallest floc sizes ($50\mu m$) close to the bed during maximum flood/ebb periods and the largest ($500\mu m$) during slack periods. We can note that flocculation starts earlier in the upper part of the water column, due to i) lower shear rate and ii) larger SSC values. Next flocs settle and accumulate close to the bed before settling in the sediment compartment.

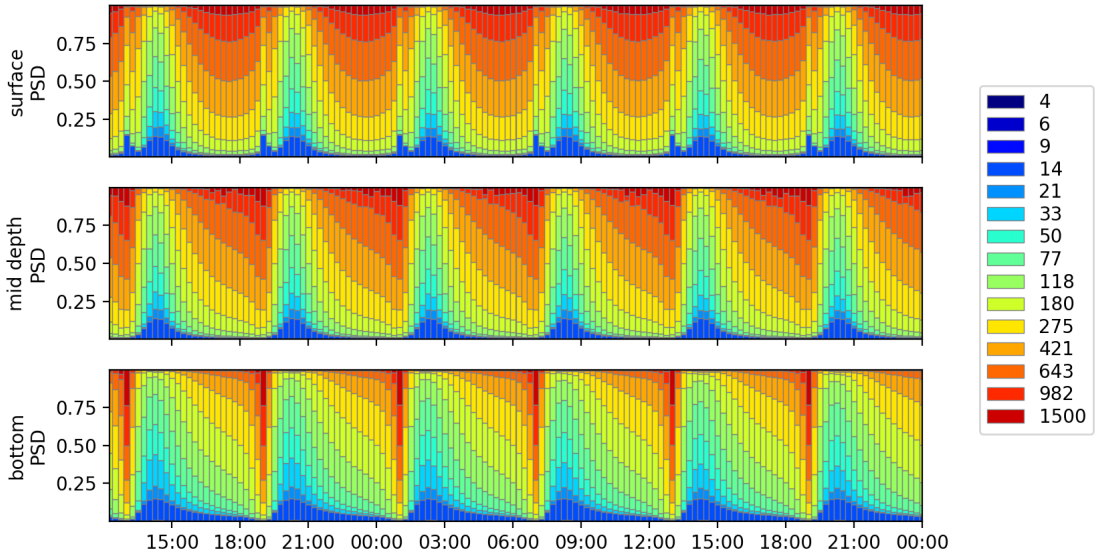
Adding differential settling aggregation

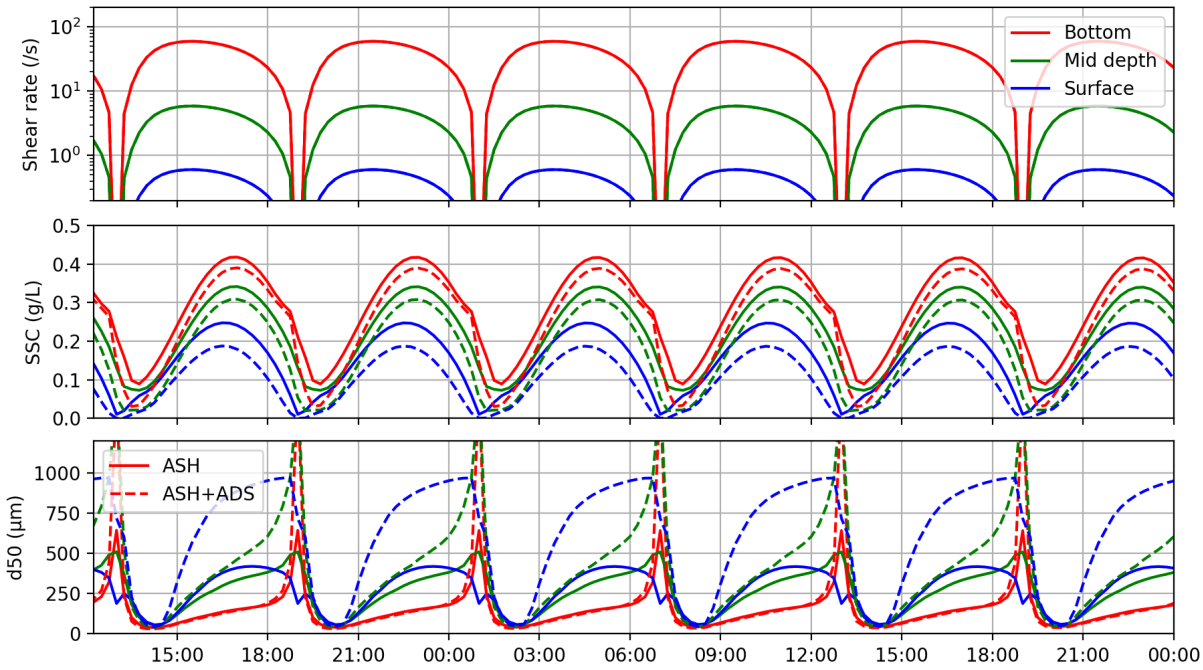
For this test, the same setup as above is used, and aggregation by differential settling is also activated (L_ADS=.TRUE.). Adding a complementary aggregation term (and a constant fragmentation term) induces a more intense flocculation, especially in the upper half of the water column, where the shear stress is less intense (D50 greater than 1mm). Close to the bed, the floc dynamics is similar to the reference, except during slack period where settling is dominant. As a consequence, large floc sizes imply lower SSC especially in the upper part of the water column.


03:30
04:30
06:00
08:30
FLOCMOD ref
FLOCMOD
Floc size distribution (g/L)
0.025
0.020
0.015
0.010
0.005
0.000
10^{-5}
10^{-4}
10^{-3}
Diameter (m)


u* (m/s)
10^{-1}
10^{-2}
10^{-3}
Shear rate (/s)
z (m)
0
−5
10^{2}
10^{1}
10^{0}
10^{-1}
SSC (g/L)
0.4
0.2
0.0
Floc d50 (µm)
500
250
0
15:00
18:00
21:00
00:00
03:00
06:00
09:00
12:00
15:00
18:00
21:00
00:00
time


Shear rate (/s)
Bottom
Mid depth
Surface
SSC (g/L)
d50 (µm)
15:00
18:00
21:00
00:00
03:00
06:00
09:00
12:00
15:00
18:00
21:00
00:00

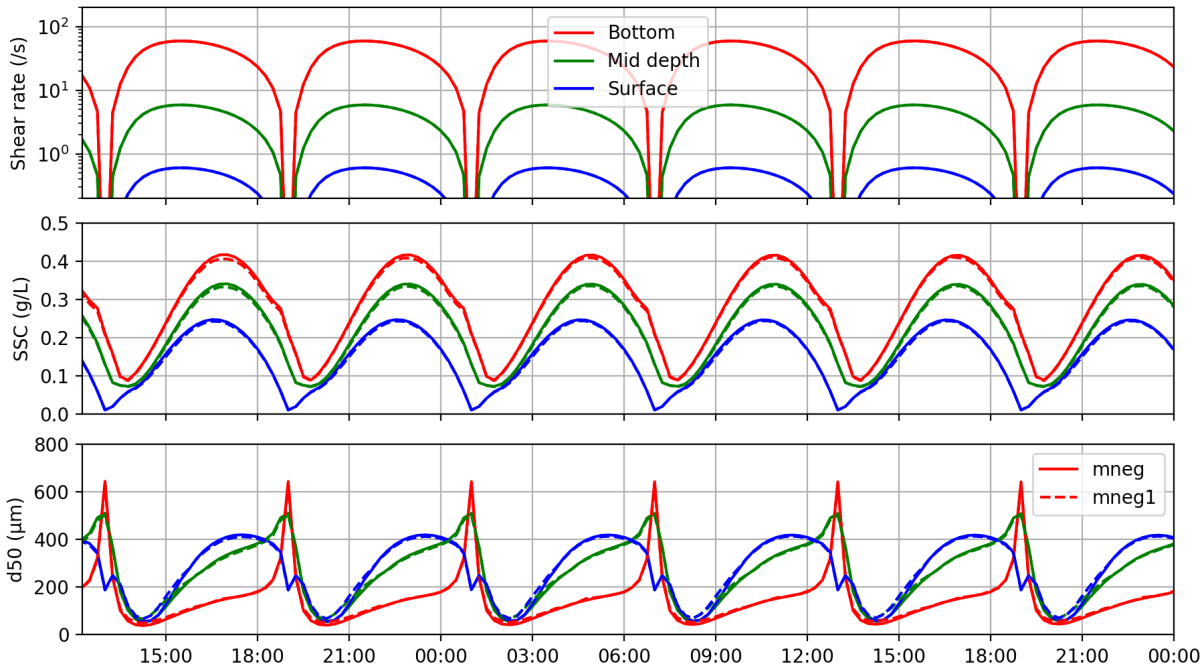

surface PSD
mid depth PSD
bottom PSD
0.75
0.50
0.25
4
6
9
14
21
33
50
77
118
180
275
421
643
982
1500
15:00
18:00
21:00
00:00
03:00
06:00
09:00
12:00
15:00
18:00
21:00
00:00



*Adding "low negative mass option" mneg_param = 0.001 g/L.**

This test case is similar to the first 1DV test case (ADS not activated), except that low negative mass is "allowed" to limit the number of sub time steps. This means that when the total negative mass is below mneg_param, negative classes have their masses set to 0 and the remaining positive classes are proportionally lowered to ensure mass conservation.

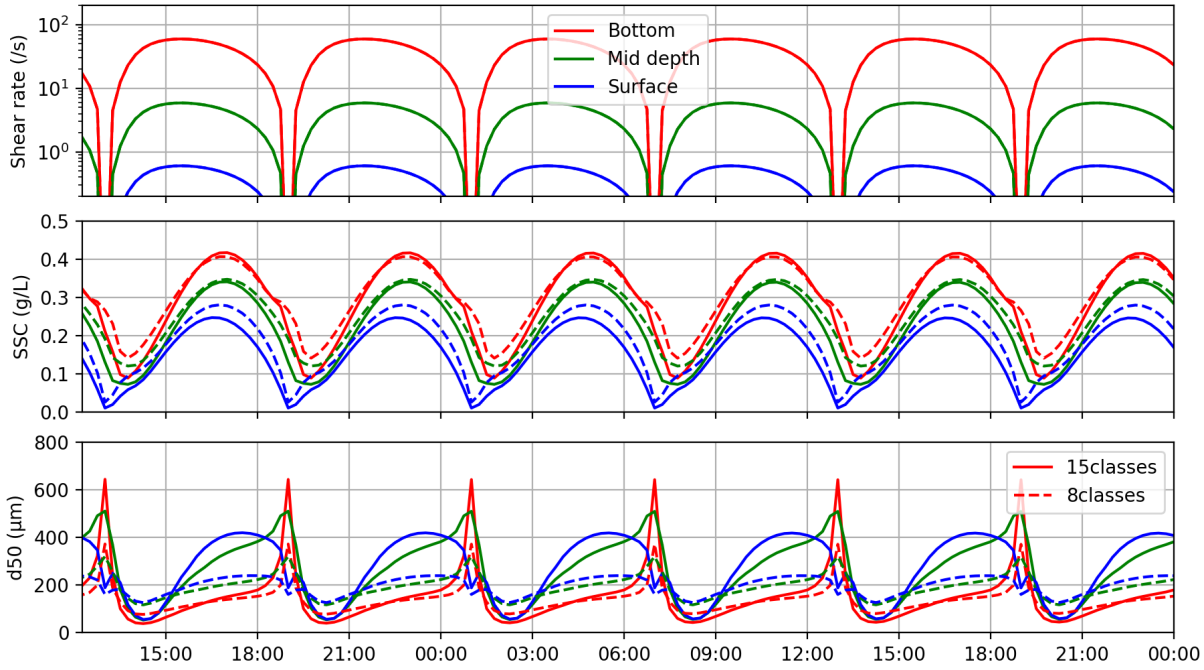
Results are very similar both in term of SSC and floc D50, hence validating the possibility to use this option to improve computation time. For this 1DV configuration, activating this option decreased the computation time of about 30%.



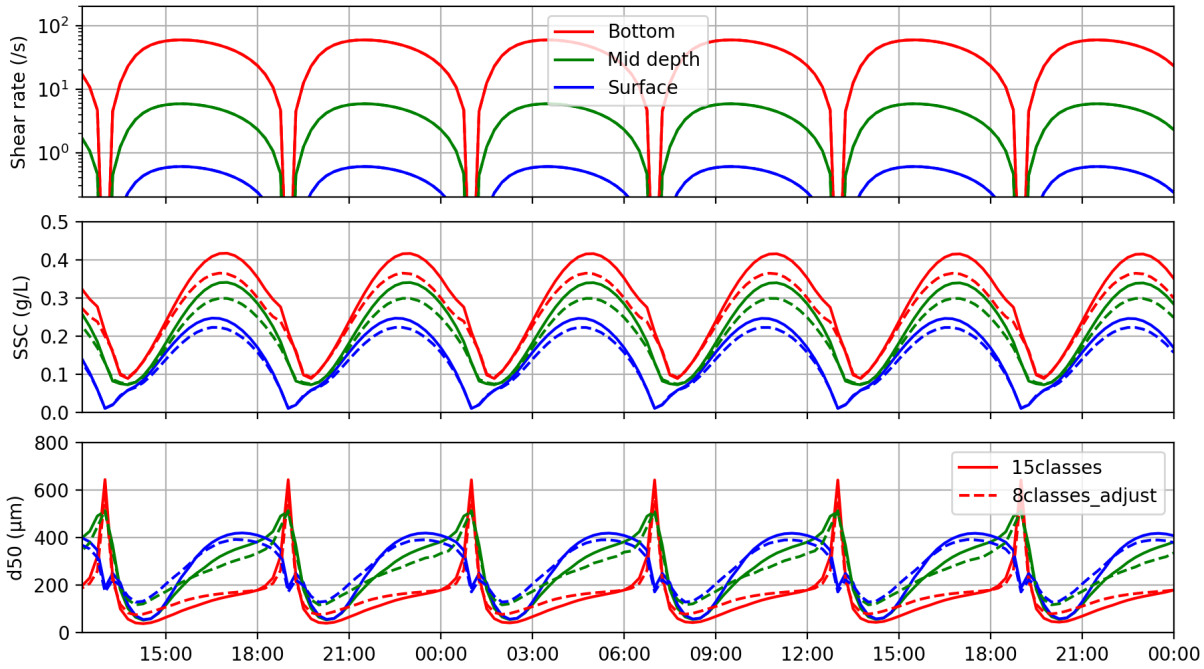
Passing from 15 classes to 8 classes

This last test concerns the number of classes to be used. The 15 original sediment classes are : [4; 6; 9; 14; 21; 33; 50; 77; 118; 180; 275; 421; 643; 982; 1500] in μm. We run exactly the same configuration but with 8 classes : [50; 77; 118; 180; 275; 421; 643; 982] in μm.

In this case, flocculation is less intense, which can be explained by a less important numerical diffusion induced by small size classes (aggregating with the largest).



Very similar results compared with our reference (15 classes) can be obtained when tuning alpha (= 0.8) and f_ero_frac (= 0.3).



Switching from 15 to 8 classes is crucial in term of computation time, saving up to 85% of computation time.

CHAPTER

NINETEEN

APPENDICES

19.1 cppdefs.h

This file defines the CPP keys that are used by the the C-preprocessor when compiling CROCO. The C-preprocessor selects the different parts of the Fortran code which needs to be compiled depending on the defined CPP options. These options are separated in two parts: the basic option keys in `cppdefs.h` and the advanced options keys in `cppdefs_dev.h`.

CPP keys define the case: test case, realistic case, as well as the numerical schemes, parameterizations, and modules used, and the forcing and boundary conditions.

- **Configuration**

CPP options	Description
BASIN	Must be defined for running the Basin example
CANYON_A	Must be defined for running the Canyon_A example
CANYON_B	Must be defined for running the Canyon_B example
EQUATOR	Must be defined for running the Equator example
GRAV_ADJ	Must be defined for running the Gravitational Adjustment example
ACOUSTIC	Must be defined for running the accoustic example
INNERSHELF	Must be defined for running the Inner Shelf example
OVERFLOW	Must be defined for running the Gravitational/Overflow example
SEAMOUNT	Must be defined for running the Seamount example
SHELFRONT	Must be defined for running the Shelf Front example
SOLITON	Must be defined for running the Equatorial Rossby Wave example
UPWELLING	Must be defined for running the Upwelling example
INTERNAL	Must be defined for running the Internal tides example
VORTEX	Must be defined for running the Baroclinic Vortex example
JET	Must be defined for running the Jet example
THACKER	Must be defined for running the Thacker example
TANK	Must be defined for running the Tank example
S2DV	Must be defined for running the S2DV example
RIP	Must be defined for running the Rip current example
SHOREFACE	Must be defined for running the Shoreface example
SWASH	Must be defined for running the Swash example
REGIONAL	Must be defined if running realistic regional simulations

- **Prallelisation**

CPP options	Description
OPENMP	Activate Open-MP parallelization protocol
MPI	Activate MPI parallelization protocol
PARALLEL_FILES	Activate parallel I/O writing
XIOS	Use external server for output
AUTOTILING	Activate subdomains partitionning optimization

- **Nesting**

CPP options	Description
AGRIF	Activate nesting capabilities (1-WAY by default)
AGRIF_2WAY	Activate 2-WAY nesting (update parent solution by child solution)

- **Open Boundary Conditions**

CPP options	Description
OBC_EAST	Open eastern boundary
OBC_WEST	Open western boundary
OBC_SOUTH	Open southern boundary
OBC_NORTH	Open northern boundary

- **Tides**

CPP options	Description
TIDES	Activate tidal forcing at open boundaries
SSH_TIDES	process and use tidal sea level data
UV_TIDES	process and use tidal current data
TIDERAMP	Apply ramping on tidal forcing (1 day) at initialization Warning! This should be undefined if restarting the model

- **Applications**

CPP options	Description
BIOLOGY	Activate biogeochemical modeling
FLOATS	Activate floats
STATIONS	Store high frequency model outputs at stations
PASSIVE_TRACER	Add a passive tracer
SEDIMENT	Activate sediment modeling
BBL	Activate bottom boundary layer parametrization

- **Grid Configuration**

CPP options	Description
CURVGRID	Activate curvilinear coordinate transformation
SPHERICAL	Activate longitude/latitude grid positioning
MASKING	Activate land masking
WET_DRY	Activate wetting-Drying scheme
NEW_S_COORD	Choose new vertical S-coordinates

- **Model Dynamics**

CPP options	Description
SOLVE3D	solve 3D primitive equations
UV_COR	Activate Coriolis terms
UV_ADV	Activate advection terms
NBQ	Activate non-boussinesq option

- **Lateral Momentum Advection**

CPP options	Description
DV_UP3	Activate 3rd-order upstream biased advection scheme
UV_HADV_UP5	Activate 5th-order upstream biased advection scheme
UV_HADV_C2	Activate 2nd-order centred advection scheme (should be used with explicit momentum mixing)
UV_HADV_C4	Activate 4th-order centred advection scheme (should be used with explicit momentum mixing)
UV_HADV_C6	Activate 6th-order centred advection scheme (should be used with explicit momentum mixing)
UV_HADV_WENO5	Activate WENO 5th-order advection scheme
UV_HADV_TVD	Activate Total Variation Diminushing scheme

- **Lateral Momentum Mixing**

CPP options	Description
UV_MIX_GEO	Activate mixing on geopotential (constant depth) surfaces
UV_MIX_S	Activate mixing on iso-sigma (constant sigma) surfaces
UV_VIS2	Activate Laplacian horizontal mixing of momentum
UV_VIS4	Activate Bilaplacian horizontal mixing of momentum
UV_VIS_SMAGO	Activate Smagorinsky parametrization of turbulent viscosity (only with UV_VIS2)
UV_VIS_SMAGO3D	Activate 3D Smagorinsky parametrization of turbulent viscosity

- **Lateral Tracer Advection**

CPP options	Description
TS_HADV_UP3	3rd-order upstream biased advection scheme
TS_HADV_RSUP3	Split and rotated 3rd-order upstream biased advection scheme
TS_HADV_UP5	5th-order upstream biased advection scheme
TS_HADV_RSUP5	Split and rotated 5th-order upstream biased advection scheme with reduced dispersion/diffustion
TS_HADV_C4	4th-order centred advection scheme
TS_HADV_C6	Activate 6th-order centred advection scheme
TS_HADV_WENO5	5th-order WENOZ quasi-monotonic advection scheme for all tracers
BIO_HADV_WENO5	5th-order WENOZ quasi-monotone advection scheme for passive tracers (including biology and sediment tracers)

- **Lateral Tracer Mixing**

CPP options	Description
TS_MIX_ISO	Activate mixing along isopycnal (isoneutral) surfaces
TS_MIX_GEO	Activate mixing along geopotential surfaces
TS_MIX_S	Activate mixing along iso-sigma surfaces
TS_DIF2	Activate Laplacian horizontal mixing of tracer
TS_DIF4	Activate Bilaplacian horizontal mixing of tracer
TS_MIX_IMP	Activate stabilizing correction of rotated diffusion (used with TS_MIX_ISO and TS_MIX_GEO)

- **Nudging**

CPP options	Description
ZNUDGING	Activate nudging layer for zeta.
M2NUDGING	Activate nudging layer for barotropic velocities.
M3NUDGING	Activate nudging layer for baroclinic velocities.
TNUDGING	Activate nudging layer for tracer.
ROBUST_DIAG	Activate strong tracer nudging in the interior for diagnostic simulations

- **Vertical Mixing**

CPP options	Description
BODYFORCE	Apply surface and bottom stresses as body-forces
ANA_VMIX	Activate analytical viscosity/diffusivity coefficients
BVF_MIXING	Activate a simple mixing scheme based on the Brunt-Väisälä frequency
LMD_MIXING	Activate Large/McWilliams/Doney mixing (turbulent closure for interior and planetary boundary layers) with following options

LMD_SKPP	Activate surface boundary layer KPP mixing
LMD_BKPP	Activate bottom boundary layer KPP mixing
LMD_RIMIX	Activate shear instability interior mixing
LMD_CONVEC	Activate convection interior mixing
LMD_DDMIX	Activate double diffusion interior mixing
LMD_NONLOCAL	Activate nonlocal transport for SKPP

GLS_MIXING	Activate Generic Length Scale scheme as implemented by Warner et al. (2005), default is k-kl (see below)
GLS_MIX2017	Activate Generic Length Scale scheme with a sligthly different implementation (under test), default is k-epsilon (see below)
GLS_KKL	Activate K-KL (K=TKE; L=length Scale) as in Mellor-Yamada 2.5 (1974)
GLS_KOMEGA	Activate K-OMEGA (OMEGA=frequency of TKE dissipation) originating from Kolmogorov (1942)
GLS_KEPSILON	Activate K-EPSILON (EPSILON=TKE dissipation) as in Jones and Launder (1972)
GLS_GEN	Activate generic model of Umlauf and Burchard (2003)

- **Equation of State**

CPP options	Description
SALINITY	Activate salinity as an active tracer
NONLIN_EOS	Activate nonlinear equation of state
SPLIT_EOS	Activate the split of the nonlinear equation of state in adiabatic and compressible parts for reduction of pressure gradient errors

- **Surface Forcing**

CPP options	Description
BULK_FLUX	Activate bulk formulation for surface heat fluxes
BULK_FAIRALL	Choose Fairall formulation (default: COAMPS formulation)
BULK_EP	Add in bulk formulation for fresh water fluxes
BULK_LW	Add in long-wave radiation feedback from model SST
BULK_SMFLUX	Add in bulk formulation for surface momentum fluxes
SST_SKIN	Activate skin sst computation (Zeng & Beljaars, 2005)
ONLINE	Read native files and perform online interpolation on ROMS grid (default cubic interpolation)
QCORRECTION	Activate linearized bulk formulation providing heat flux correction dQdSST for nudging towards model SST
SFLX_CORR	Activate freshwater flux correction around model SSS
ANA_DIURNAL_SW	Activate analytical diurnal modulation of short wave radiations (only appropriate if there is no diurnal cycle in data)

- **Coupling**

CPP options	Description
OW_COUPLING	Activate Ocean-Wave coupling
OA_COUPLING	Activate Ocean-Atmosphere coupling
OA_MCT	Use OASIS-MCT for coupling

- **Sponge Layer**

CPP options	Description
SPONGE	Activate areas of enhanced viscosity and diffusivity near lateral open boundaries.
SPONGE_GRID	Automatic setting of the sponge width and value
SPONGE_DIF2	Sponge on tracers (default)
SPONGE_VIS2	Sponge on momentum (default)
SPONGE_SED	Sponge on sediment (default)

- **Lateral forcing**

CPP options	Description
CLIMATOLOGY	Activate processing of 2D/3D climatological data for nudging layers and open boundary forcing
ZCLIMATOLOGY	Activate processing of sea level
M2CLIMATOLOGY	Activate processing of barotropic velocities
M3CLIMATOLOGY	Activate processing of baroclinic velocities
TCLIMATOLOGY	Activate processing of tracers
ZNUDGING	Activate nudging layer for sea level
M2NUDGING	Activate nudging layer for barotropic velocities
M3NUDGING	Activate nudging layer for baroclinic velocities
TNUDGING	Activate nudging layer for tracers
ROBUST_DIAG	Activate nudging over the whole domain
FRC_BRY	Activate processing of 1D/2D climatological or simulation/reanalysis data at open boundary points
Z_FRC_BRY	Activate open boundary forcing for sea level
M2_FRC_BRY	Activate open boundary forcing for barotropic velocities
M3_FRC_BRY	Activate open boundary forcing for baroclinic velocities
T_FRC_BRY	Activate open boundary forcing for tracers

- **Bottom Forcing**

CPP options	Description
ANA_BSFLUX	Activate analytical bottom salinity flux (generally 0)
ANA_BTFLUX	Activate analytical bottom temperature flux (generally 0)

- **Point Sources - Rivers**

CPP options	Description
PSOURCE	Activate point sources (rivers)
ANA_PSOURCE	use analytical vertical profiles for point sources (set in set_global_definitions.h)
PSOURCE_NCFILE	Read variable river transports in netcdf file
PSOURCE_NCFILE_TS	Read variable river concentration in netcdf file

- **Open boundary conditions II**

CPP options	Description
OBC_VOLCONS	Activate mass conservation enforcement at open boundaries (with OBC_M2ORLANSKI)
OBC_M2SPECIFIED	Activate specified open boundary conditions for barotropic velocities
OBC_M2ORLANSKI	Activate radiative open boundary conditions for barotropic velocities
OBC_M2FLATHER	Activate Flather open boundary conditions for barotropic velocities
OBC_M2CHARACT	Activate open boundary conditions based on characteristic methods barotropic velocities
OBC_M3SPECIFIED	Activate specified open boundary conditions for baroclinic velocities
OBC_M3ORLANSKI	Activate radiative open boundary conditions for baroclinic velocities
OBC_TSPECIFIED	Activate specified open boundary conditions for tracers
OBC_TORLANSKI	Activate radiative open boundary conditions for tracers
OBC_TUPWIND	Activate upwind open boundary conditions for tracers

- **I/O - Diagnostics**

CPP options	Description
AVERAGES	Process and output time-averaged data
AVERAGES_K	Process and output time-averaged vertical mixing
DIAGNOSTICS_TS	Store and output budget terms of the tracer equations
DIAGNOSTICS_TS_ADV	Choose advection rather than transport formulation for tracer budgets
DIAGNOSTICS_TS_MLD	Integrate tracer budgets over the mixed-layer depth
DIAGNOSTICS_UV	Store and output budget terms of the momentum equations
XIOS	Use XIOS IO server

- **Biogeochemical models**

CPP options	Description
PISCES	Activate 24-component PISCES biogeochemical model
BIO_NChlPZD	Activate 5-component NPZD type model
BIO_N2PZD2	Activate 7-component NPZD type model
BIO_BioEBUS	Activate 12-component NPZD type model

- **Sediment Dynamics**

CPP options	Description
ANA_SEDIMENT	Set analytical sediment ripple and bed parameters
ANA_WWAVE	Analytical (constant) wave (hs,Tp,Dir) values.
SUSPLOAD	Activate suspended load transport
BEDLOAD	Activate bedload transport
MORPHODYN	Activate morphodynamics
ANA_BPFLUX	Set kinematic bottom flux of sediment tracer (if different from 0)
SLOPE_NEMETH	Nemeth formulation for avalanching (Nemeth et al, 2006)
SLOPE_LESSER	Lesser formulation for avalanching (Lesser et al, 2004)
BED-LOAD_SOULSBY	Soulsby formulation for bedload (Soulsby, R.L. and J.S. Damgaard, 2005)
BEDLOAD_MPM	Meyer-Peter-Muller formulation for bedload (Meyer-Peter, E. and R. Muller, 1948)

- **Bottom Boundary Layer**

CPP options	Description
ANA_WWAVE	Set analytical wave forcing
ANA_BSEDIM	Set analytical bed parameters (if SEDIMENT is un-defined)
Z0_BL	Compute bedload roughness for ripple predictor and sediment purposes
Z0_RIP	Determine bedform roughness ripple height and ripple length for sandy beds
Z0_BIO	Determine (biogenic) bedform roughness ripple height and ripple length for silty beds

- **Wave-Current interactions**

CPP options	Description
MRL_WCI	Set wave-current interaction model
ANA_WWAVE	Analytical values for waves
WAVE_OFFLINE	Set-wave offline (read from file) forcing
WKB_WWAVE	Set WKB wave model
OW_COUPLING	Set wave coupling for WWIII (to install separately)
MRL_CEW	Set current feedback on waves
WKB_OBC_WEST	Set East/West/North/South offshore forcing for WKB model
ANA_BRY_WKB	Analytical boundary wave foring (from croco.in)
WAVE_BREAK_CT93	Thornton and Guza (1983, JGR) wave breaking (for WKB)
WAVE_BREAK_TG86	Church and Thornton (1993, Coastal Eng.) wave breaking (for WKB)
WAVE_BREAK_TG86A	Another Church and Thornton formulation
WAVE_ROLLER	Set wave roller model
WAVE_STREAMING	Set bottom wave streaming
WAVE_FRICTION	Set bottom friction in WKB model (used for streaming)
WAVE_RAMP	Set wave forcing ramp using wave_ramp parameter

19.2 croco.in

KEYWORD	DESCRIPTION
title	Configuration name
time_stepping	NTIMES : Number of time-steps required for the simulation dt : Baroclinic time step [in s] NDTFAST : Number of barotropic time-steps between each baroclinic time step. For 2D configurations, ndtfast should be unity NINFO : Number of time-steps between printing of information to standard output
time_stepping_nbq	NDTNBQ CSOUND_NBQ VISC2_NBQ
S-coord	THETA_S: S-coordinate surface control parameter THETA_B: S-coordinate bottom control parameter Hc(m): Width of surface or bottom boundary layer in which higher vertical resolution is required during stretching
run_start_date	run start date (used with `USE_CALENDAR`)
run_end_date	run end date (used with `USE_CALENDAR`)
output_time_steps	DT_HIS(H) DT_AVG(H) DT_RST(H)
grid	grid filename
forcing	forcing filename
bulk_forcing	bulk forcing filename (used with `BULK_FLUX`)
climatology	climatology filename (boundary and nudging, used with `CLIMATOLOGY`)
boundary	boundary filename (used with `FRC_BRY``)
initial	NRREC: Switch to indicate start or re-start from a previous solution. nrrec is the time index of the initial or re-start NetCDF file assigned for initialization. If nrrec is negative (say nrrec = -1), the model will start from the most recent time record. That is, the initialization record is assigned internally. filename: Name of file containing the initial state.

Continued on next page

Table 1 – continued from previous page

KEYWORD	DESCRIPTION
restart	NRST: Number of time-steps between writing of re-start fields NRPFRST 0: write several records every NRST time steps >0: create more than one file (with sequential numbers) and write NRPRST records per file -1: overwrite record every NRST time steps filename: name of restart file
history	LDEFHIS: flag (T/F) for writing history files NWRT: Number of time-steps between writing of history fields NRPFHIS: 0: write several records every NWRT time steps >0: create more than one file (with sequential numbers) and write NRPHIS records per file -1: overwrite record every NWRT time steps filename: Name of history file
averages	NTSAVG: Starting timestep for the accumulation of output time-averaged data. For instance, you might want to average over the last day of a thirty-day run. NAVG: Number of time-steps between writing of averaged fields NRPFAVG: 0: write several records every NAVG time steps >0: create more than one file (with sequential numbers) and write NRPFAVG records per file -1: overwrite record every NAVG time steps filename: Name of average file
primary_history_fields	Flags for writing primary variables in history NetCDF file
auxiliary_history_fields	Flags for writing auxiliary variables in history NetCDF file
primary_averages	Flags for writing primary variables in average NetCDF file
auxiliary_averages	Flags for writing auxiliary variables in average NetCDF file
rho0	Mean density used in the Boussinesq equation.
lateral_visc	VISC2: Laplaplacian background viscosity in m2/s (with UV_VIS2 CPP option) VISC4: Bilaplacian background viscosity in m4/s (with UV_VIS4 CPP option)

Continued on next page

Table 1 – continued from previous page

KEYWORD	DESCRIPTION
tracer_diff2	TNU2(1:NT): Laplacian background diffusivity in m2/s for each tracer (with TS_DIF2 CPP option)
tracer_diff4	TNU4(1:NT): Laplacian background diffusivity in m4/s for each tracer (with TS_DIF4 CPP option)
vertical_mixing	Constant vertical viscosity coefficient in m2/s for analytical vertical mixing scheme (with ANA_VMIX CPP option)
bottom_drag	RDRG [m/s]: Drag coefficient for linear bottom stress formulation RDRG2: Drag coefficient for constant quadratic bottom stress formulation Zob [m]: Roughness length for Von-Karman quadratic bottom stress formulation Cdb_min: Minimum value of drag coefficient for Von-Karman quadratic bottom stress formulation Cdb_max: Maximum value of drag coefficient for Von-Karman quadratic bottom stress formulation.
gamma2	Free- or partial- or no-slip wall boundary condition. 1 means free slip conditions are used.
sponge	sponge parameters are only needed if SPONGE_GRID is undefined in set_global_definitions.h; otherwise, these parameters are assigned internally. X_SPONGE [m]: width of sponge layers V_SPONGE [m2/s]: viscosity/diffusivity values in sponge layers. These values follow a cosine profile from zero interior value to V_SPONGE at the boundary.

Continued on next page

Table 1 – continued from previous page

KEYWORD	DESCRIPTION
nudg_cof	TauT_in [days]: Short nudging time scale used in radiative open boundary conditions for tracer signal propagating inward the computational domain. This coefficient is used at boundary points and imposes strong nudging towards external data TauT_out [days]: Long nudging time scale used in radiative open boundary conditions for tracer signal propagating outward the computational domain. This coefficient is used at boundary points and imposes mild nudging towards external data. If CLIMATOLOGY is defined, it is also used in nudging layers with gradual decrease (cosine profile) from the open boundary to the inner border of the nudging layer. TauM_in [days]: Same as above, but for momentum equations TauM_out [days]: Same as above, but for momentum equations
diagnostics	ldefdia: flag that activates the storage of the instantaneous tracer budget terms in a diagnostic file nwrtdia: Number of time-steps between writing of diagnostic fields nrpfdia: 0: write several records every NWRTDIA time steps >0: create more than one file (with sequential numbers) and write NRPFDIA records per file -1: overwrite record every NWRTDIA time steps filename: Name of instantaneous tracer diagnostic file

Continued on next page

Table 1 – continued from previous page

KEYWORD	DESCRIPTION
diag_avg	ldefdia_avg: flag that activates the storage of averaged tracer budget terms in a diagnostic file ntsdia_avg: Starting timestep for the accumulation of output time-averaged data. For instance, you might want to average over the last day of a thirty-day run. nwrtdia_avg: Number of time-steps between writing of averaged diagnostic fields nprfdia_avg: 0: write several records every NWRTDIA_AVG time steps >0: create more than one file (with sequential numbers) and write NRPFDIA_AVG records per file -1: overwrite record every NWRTDIA_AVG time steps filename: Name of average tracer diagnostic file
diag3D_history_fields	flag to select which tracer equation (temp, salt, etc . . .) to store in diagnostic file. These terms are 3D.
diag2D_history_fields	flags to select which tracer equation integrated over the mixed layer depth (cf DIAGNOSTICS_TS_MLD) to store in diagnostic file. These terms are 2D.
diag3D_average_fields	same as diag3D_history _fields but for averaged fields
diag2D_average_fields	same as diag2D_history _fields but for averaged fields
diagnosticsM	Same format as diagnostics but for momentum equations ldefdiaM: nwrtdiaM: nrpfdiaM: filename:
diagM_avg	Same format as diag_avg but for momentum equations
diagM_history_fields	flag to select which momentum equation (u,v) to store in diagnostic file. These terms are 3D.
diagM_average _fields	same as diagM_history _fields but for averaged fields
diagnosticsM_bio	Same format as diagnostics but for biogeochemical budget terms (other than advection/diffusion)
diagbio_avg	Same format as diag_avg but for biogeochemical budget terms (other than advection/diffusion)

Continued on next page

Table 1 – continued from previous page

KEYWORD	DESCRIPTION
diagbioFlux_history_fields	Flag (T or F) to select which biogechemical tracer flux to store in diagnostic file. These terms are 3D. (For NPZD type model)
diagbioVSink_history_fields	Flag (T or F) to select which biogechemical tracer sinking flux equation to store in diagnostic file These terms are 3D. This is for NPZD type model(BIO_NChlPZD, BIO_N2ChlPZD2 and BIO_BioEBUS), you need to follow the biogechemical tracers order.
diagbioGasExc_history_fields	Flag (T or F) to select which biogechemical tracer Gas exchange flux equation to store in diagnostic file. These terms are 2D.
diagbioFlux_average_fields	Same as above but averaged
diagbioVSink_average_fields	Same as above but averaged
diagbioGasExc_average_fields	Same as above but averaged
biology	Name of file containing the Iron dust forcing used in the PISCES biogeochemical model
sediments	input file: sediment parameters input file
sediment_history_fields	Flags for storing sediment fields in history file bed_thick:Thickness of sediment bed layer (m) bed_poros: Porosity of sediment bed layer (no unit) bed_fra(sand,silt): Volume fraction of sand/silt in bed layer (no unit)
bbl_history_fieldsi	Flags for storing bbl fields in history file Abed: Bed wave excursion amplitude (m) Hripple: Bed ripple length (m) Lripple: Bed ripple length (m) Zbnot: Physical hydraulic bottom roughness (m) Zbapp: Apparent hydraulic bottom roughness (m) Bostrw: Wave-induced kinematic bottom stress (m)
floats	Lagrangian floats application. Same format as diagnostics LDEFFLT NFLT NRPFFLT inpname, hisname
floats_fields	Type of fields computed for each lagrangian floats

Continued on next page

Table 1 – continued from previous page

KEYWORD	DESCRIPTION
station_fields	Fixed station application. Same format as diagnostics LDEFSTA NSTA NRPFSTA inpname, hisname
psource	Nsrc: point source number Isrc: I point source indice Jsrc: J point source indice Dsrc: Direction of point source flow (u=0,v=1) Qbar [m3/s]: Total transport at point source Lsrc: Logical switch for type of tracer to apply Tsrc: Tracer value
psource_ncfile	Nsrc: point source number Isrc: I point source indice Jsrc: J point source indice Dsrc: Direction of point source flow (u=0,v=1) qbardir: Orientation: South=0 or North=0, East=0 or West=1 Lsrc: Logical switch for type of tracer to apply Tsrc: Tracer value in case of analytical value [#undef PSOURCE_NCFILE_TS] runoff file name: Input netCDF runoff file

19.3 Comparison of ROMS and CROCO versions

Models	CROCO	ROMS-UCLA	ROMS-Rutgers / COASWT
Origin	UCLA-IRD-INRIA-IFREMER-SHOM-CNRS	UCLA	UCLA-Rutgers-USGS
Maintenance	IRD-INRIA-IFREMER-SHOM-CNRS	UCLA	Rutgers-USGS
Realm	Europe-World	US West Coast	US East Coast
Introductory year	1999 (AGRIF) 2016 (CROCO)	2002	1998

CODE FEATURES

Models	CROCO	ROMS-UCLA	ROMS-Rutgers / COASWT
Parallelization	MPI or OpenMP (Hybrid branch exists)	Hybrid MPI-OpenMP	MPI or OpenMP
Nesting	Online at barotropic level	Off-line (On-line at baroclinic level and not yet operational)	Off-line
Data assimilation	3DVAR	3DVAR	4DVAR
Wave-current interact.	McWilliams et al. 2004	McWilliams et al. 2004	Mellor (2003) and McWilliams et al. (2004)
Air-sea coupling	OASIS-MCT	Home made	MCT
Sediment Dynamics	Blaas et al. (2007), MUSTANG	Blaas et al. (2007)	Blaas et al. (2007), Warner et al. (2008)
Biogeochemistry	NPZD Gruber et al. (2006), PISCES	NPZD Gruber et al. (2006)	EcoSim, NEMURO, NPZD Franks, NPZD Powell, Fennel
Sea ice	none	none	Budgell (2005)
Vertical mixing	KPP, GLS	KPP, GLS	KPP, GLS
Wetting-Drying	Warner et al. (2013)	none	Warner et al. (2013)

TIME STEPPING

Models	CROCO	ROMS-UCLA	ROMS-Rutgers / COASWT
2D momentum	Generalized FB AB3-AM4	Generalized FB AB3-AM4	LF-AM3 with FB feedback
3D momentum	LF-AM3	LF-AM3	AB3
Tracers	LF-AM3 with stabi lizing correction for isopycnal hyperdiffusion	LF-AM3 with stabi lizing correction for isopycnal hyperdiffusion	LF-TR with explicit geopotential diffusion (no stabilizing correction: strong stabil ity constraint)
Internal waves	LF-AM3 with FB feedback	LF-AM3 with FB feedback	Generalized FB (AB3-TR)
Coupling stage	Predictor	Corrector	

STABILITY CONSTRAINTS (Max Courant number)

Models	CROCO	ROMS-UCLA	ROMS-Rutgers / COASWT
2D	1.78	1.78	1.85
3D advection	1.58	1.58	0.72
Coriolis	1.58	1.58	0.72
Internal waves	1.85	1.85	1.14