NEMO ocean engine

– version 3.1 –

Gurvan Madec

gurvan.madec@locean-ipsl.umpc.fr

Laboratoire d'Océanographie et du Climat: Expérimentation et Approches Numériques

May 2008

Note du Pôle de modélisation de l'Institut Pierre-Simon Laplace No 27

ISSN No 1288-1619.

Table des matières

Abstract / Résumé

The ocean engine of NEMO (Nucleus for European Modelling of the Ocean) is a primitive equation model adapted to regional and global ocean circulation problems. It is intended to be a flexible tool for studying the ocean and its interactions with the others components of the earth climate system (atmosphere, sea-ice, biogeochemical tracers, ...) over a wide range of space and time scales. Prognostic variables are the three-dimensional velocity field, a linear or non-linear sea surface height, the temperature and the salinity. In the horizontal direction, the model uses a curvilinear orthogonal grid and in the vertical direction, a full or partial step z-coordinate, or s-coordinate, or a mixture of the two. The distribution of variables is a three-dimensional Arakawa C-type grid. Various physical choices are available to describe ocean physics, including TKE and KPP vertical physics. Within NEMO, the ocean is interfaced with a sea-ice model (LIM v2 and v3), passive tracer and biogeochemical models (TOP) and, via the OASIS coupler, with several atmospheric general circulation models.

Le moteur océanique de NEMO (Nucleus for European Modelling of the Ocean) est un modèle aux équations primitives de la circulation océanique régionale et globale. Il se veut un outil flexible pour étudier sur un vaste spectre spatiotemporel l'océan et ses interactions avec les autres composantes du système climatique terrestre (atmosphère, glace de mer, traceurs biogéochimiques...). Les variables pronostiques sont le champ tridimensionnel de vitesse, une hauteur de la mer linéaire ou non, la temperature et la salinité. La distribution des variables se fait sur une grille C d'Arakawa tridimensionnelle utilisant une coordonnée verticale z à niveaux entiers ou partiels, ou une coordonnée s, ou encore une combinaison des deux. Différents choix sont proposés pour décrire la physique océanique, incluant notamment des physiques verticales TKE et KPP. A travers l'infrastructure NEMO, l'océan est interfacé avec un modèle de glace de mer, des modèles biogéochimiques et de traceur passif, et, via le coupleur OASIS, à plusieurs modèles de circulation générale atmosphérique.

Disclaimer

NEMO reference in papers and other publications is as follows :

Madec, G., 2008 : NEMO ocean engine. *Note du Pôle de modélisation*, Institut Pierre-Simon Laplace (IPSL), France, No 27, ISSN No 1288-1619.

Additional information can be found on http ://www.nemo-ocean.eu/

Introduction

The Nucleus for European Modelling of the Ocean (*NEMO*) is a framework of ocean related engines, namely OPA[1] for the ocean dynamics and thermodynamics, LIM[2] for the sea-ice dynamics and thermodynamics, TOP[3] for the biogeochemistry (both transport (TRP) and sources minus sinks (LOBSTER, PISCES)[4]. It is intended to be a flexible tool for studying the ocean and its interactions with the other components of the earth climate system (atmosphere, sea-ice, biogeochemical tracers, ...) over a wide range of space and time scales. This documentation provides information about the physics represented by the ocean component of *NEMO* and the rationale for the choice of numerical schemes and the model design. More specific information about running the model on different computers, or how to set up a configuration, are found on the *NEMO* web site (www.locean-ipsl.upmc.fr/NEMO).

The ocean component of *NEMO* has been developed from the OPA model, release 8.2, described in **?**. This model has been used for a wide range of applications, both regional or global, as a forced ocean model and as a model coupled with the atmosphere. A complete list of references is found on the *NEMO* web site.

This manual is organised in as follows. Chapter 2 presents the model basics, $i.e.$ the equations and their assumptions, the vertical coordinates used, and the subgrid scale physics. This part deals with the continuous equations of the model (primitive equations, with potential temperature, salinity and an equation of state). The equations are written in a curvilinear coordinate system, with a choice of vertical coordinates (z or s, with the rescaled height coordinate formulation *z**, or *s**). Momentum equations are formulated in the vector invariant form or in the flux form. Dimensional units in the meter, kilogram,

[1]OPA = Océan PArallélisé

[2]LIM= Louvain)la-neuve Ice Model

[3]TOP = Tracer in the Ocean Paradigm

[4]Both LOBSTER and PISCES are not acronyms just name

second (MKS) international system are used throughout.

The following chapters deal with the discrete equations. Chapter 3 presents the space and time domain. The model is discretised on a staggered grid (Arakawa C grid) with masking of land areas and uses a Leap-frog environment for time-stepping. Vertical discretisation used depends on both how the bottom topography is represented and whether the free surface is linear or not. Full step or partial step z-coordinate or s- (terrain-following) coordinate is used with linear free surface (level position are then fixed in time). In non-linear free surface, the corresponding rescaled height coordinate formulation (*z** or *s**) is used (the level position then vary in time as a function of the sea surface heigh). The following two chapters (4 and 5) describe the discretisation of the prognostic equations for the active tracers and the momentum. Explicit, split-explicit and implicit free surface formulations are implemented as well as rigid-lid case. A number of numerical schemes are available for momentum advection, for the computation of the pressure gradients, as well as for the advection of tracers (second or higher order advection schemes, including positive ones).

Surface boundary conditions (chapter 6) can be implemented as prescribed fluxes, or bulk formulations for the surface fluxes (wind stress, heat, freshwater). The model allows penetration of solar radiation There is an optional geothermal heating at the ocean bottom. Within the *NEMO* system the ocean model is interactively coupled with a sea ice model (LIM) and with biogeochemistry models (PISCES, LOBSTER). Interactive coupling to Atmospheric models is possible via the OASIS coupler [**?**].

Other model characteristics are the lateral boundary conditions (chapter 7). Global configurations of the model make use of the ORCA tripolar grid, with special north fold boundary condition. Free-slip or no-slip boundary conditions are allowed at land boundaries. Closed basin geometries as well as periodic domains and open boundary conditions are possible.

Physical parameterisations are described in chapters 8 and 9. The model includes an implicit treatment of vertical viscosity and diffusivity. The lateral Laplacian and biharmonic viscosity and diffusion can be rotated following a geopotential or neutral direction. There is an optional eddy induced velocity [**?**] with a space and time variable coefficient **?**. The model has vertical harmonic viscosity and diffusion with a space and time variable coefficient, with options to compute the coefficients with **?**, **?**, or **?** mixing schemes.

Specific online diagnostics (not documented yet) are available in the model : output of all the tendencies of the momentum and tracers equations, output of tracers tendencies averaged over the time evolving mixed layer.

The model is implemented in FORTRAN 90, with preprocessing (C-pre-processor). It runs under UNIX. It is optimized for vector computers and parallelised by domain decomposition with MPI. All input and output is done in NetCDF (Network Common Data Format) with a optional direct access format for output. To ensure the clarity and readability of the code it is necessary to follow coding rules. The coding rules for OPA include conventions for naming variables, with different starting letters for different types of variables (real, integer, parameter. . .). Those rules are presented in a document available on the *NEMO* web site.

The model is organized with a high internal modularity based on physics. For example, each trend ($i.e.$, a term in the RHS of the prognostic equation) for momentum and tracers is computed in a dedicated module. To make it easier for the user to find his way around the code, the module names follow a three-letter rule. For example, *tradmp.F90* is a module related to the TRAcers equation, computing the DaMPing. The complete list of module names is presented in Appendix **??**. Furthermore, modules are organized in a few directories that correspond to their category, as indicated by the first three letters of their name.

The manual mirrors the organization of the model. After the presentation of the continuous equations (Chapter 2), the following chapters refer to specific terms of the equations each associated with a group of modules .

Chapter 3	DOM	model DOMain
Chapter 4	TRA	TRAcer equations (potential temperature and salinity)
Chapter 5	DYN	DYNamic equations (momentum)
Chapter 6	SBC	Surface Boundary Conditions
Chapter 7	LBC	Lateral Boundary Conditions
Chapter 8	LDF	Lateral DiFfusion (parameterisations)
Chapter 9	ZDF	Vertical DiFfusion
Chapter 10	...	Miscellaneous topics

In the current release (v3.0), the LBC directory does not yet exist. When created LBC will contain the OBC directory (Open Boundary Condition), and the *lbclnk.F90, mppini.F90* and *lib_mpp.F90* modules.

Nota Bene :

OPA, like all research tools, is in perpetual evolution. The present document describes the OPA version include in the release 3.0 of NEMO. This release differs significantly from version 8, documented in **?**. The main modifications are :
(1) transition to full native FORTRAN 90, deep code restructuring and drastic reduction of CPP keys ;
(2) introduction of partial step representation of bottom topography [**?**] ;
(3) partial reactivation of a terrain-following vertical coordinate (s- and hybrid s-z) with the addition of several options for pressure gradient computation [5] ;
(4) more choices for the treatment of the free surface : full explicit, split-explicit , filtered and rigid-lid ;
(5) non linear free surface option (associated with the rescaled height coordinate *z** or

[5]Partial support of s-coordinate : there is presently no support for neutral physics in s- coordinate and for the new options for horizontal pressure gradient computation with a non-linear equation of state.

*s**) ;
(6) additional schemes for vector and flux forms of the momentum advection ;
(7) additional advection schemes for tracers ;
(8) implementation of the AGRIF package (Adaptative Grid Refinement in FORTRAN) [**?**] ;
(9) online diagnostics : tracers trend in the mixed layer and vorticity balance ;
(10) rewriting of the I/O management ;
(11) OASIS 3 and 4 couplers interfacing with atmospheric global circulation models.
(12) surface module (SBC) that simplify the way the ocean is forced and include two bulk formulea (CLIO and CORE)
(13) introduction of LIM 3, the new Louvain-la-Neuve sea-ice model (C-grid rheology and new thermodynamics including bulk ice salinity) [**?**]

In addition, several minor modifications in the coding have been introduced with the constant concern of improving performance on both scalar and vector computers.

Model basics

Contents

2.1 Primitive Equations

2.1.1 Vector Invariant Formulation

The ocean is a fluid that can be described to a good approximation by the primitive equations, $i.e.$ the Navier-Stokes equations along with a nonlinear equation of state which couples the two active tracers (temperature and salinity) to the fluid velocity, plus the following additional assumptions made from scale considerations :

(1) spherical earth approximation : the geopotential surfaces are assumed to be spheres so that gravity (local vertical) is parallel to the earth's radius

(2) thin-shell approximation : the ocean depth is neglected compared to the earth's radius

(3) 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

(4) Boussinesq hypothesis : density variations are neglected except in their contribution to the buoyancy force

(5) Hydrostatic hypothesis : the vertical momentum equation is reduced to a balance between the vertical pressure gradient and the buoyancy force (this removes convective processes from the initial Navier-Stokes equations and so convective processes must be parameterized instead)

(6) Incompressibility hypothesis : the three dimensional divergence of the velocity vector is assumed to be zero.

Because the gravitational force is so dominant in the equations of large-scale motions, it is useful to choose an orthogonal set of unit vectors (**i**,**j**,**k**) linked to the earth such that **k** is the local upward vector and (**i**,**j**) are two vectors orthogonal to **k**, $i.e.$ tangent to the geopotential surfaces. Let us define the following variables : **U** the vector velocity, $\mathbf{U} = \mathbf{U}_h + w\,\mathbf{k}$ (the subscript h denotes the local horizontal vector, $i.e.$ over the (**i**,**j**) plane), T the potential temperature, S the salinity, ρ the *in situ* density. The vector invariant form of the primitive equations in the (**i**,**j**,**k**) vector system provides the following six equations (namely the momentum balance, the hydrostatic equilibrium, the incompressibility equation, the heat and salt conservation equations and an equation of state) :

$$\frac{\partial \mathbf{U}_h}{\partial t} = -\left[(\nabla \times \mathbf{U}) \times \mathbf{U} + \frac{1}{2}\nabla\left(\mathbf{U}^2\right)\right]_h - f\;\mathbf{k} \times \mathbf{U}_h - \frac{1}{\rho_o}\nabla_h p + \mathbf{D^U} + \mathbf{F^U} \tag{2.1a}$$

$$\frac{\partial p}{\partial z} = -\rho\, g \tag{2.1b}$$

$$\nabla \cdot \mathbf{U} = 0 \tag{2.1c}$$

$$\frac{\partial T}{\partial t} = -\nabla \cdot (T\,\mathbf{U}) + D^T + F^T \tag{2.1d}$$

$$\frac{\partial S}{\partial t} = -\nabla \cdot (S\,\mathbf{U}) + D^S + F^S \tag{2.1e}$$

$$\rho = \rho\,(T, S, p) \tag{2.1f}$$

where ∇ is the generalised derivative vector operator in $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ directions, t is the time, z is the vertical coordinate, ρ is the *in situ* density given by the equation of state (2.1f), ρ_o is a reference density, p the pressure, $f = 2\mathbf{\Omega} \cdot \mathbf{k}$ is the Coriolis acceleration (where $\mathbf{\Omega}$ is the Earth's angular velocity vector), and g is the gravitational acceleration. $\mathbf{D^U}$, D^T and D^S are the parameterisations of small-scale physics for momentum, temperature and salinity, and $\mathbf{F^U}$, F^T and F^S surface forcing terms. Their nature and formulation are discussed in §2.6 and page §2.1.2.

.

2.1.2 Boundary Conditions

An ocean is bounded by complex coastlines, bottom topography at its base and an air-sea or ice-sea interface at its top. These boundaries can be defined by two surfaces, $z = -H(i,j)$ and $z = \eta(i,j,k,t)$, where H is the depth of the ocean bottom and η is the height of the sea surface. Both H and η are usually referenced to a given surface, $z = 0$, chosen as a mean sea surface (Fig. 2.1.2). Through these two boundaries, the ocean can exchange fluxes of heat, fresh water, salt, and momentum with the solid earth, the continental margins, the sea ice and the atmosphere. However, some of these fluxes are so weak that even on climatic time scales of thousands of years they can be neglected. In the following, we briefly review the fluxes exchanged at the interfaces between the ocean and the other components of the earth system.

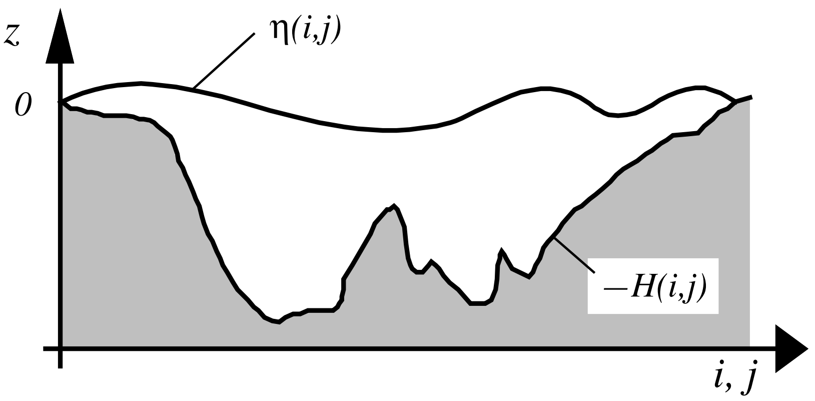


FIG. 2.1 – The ocean is bounded by two surfaces, $z = -H(i,j)$ and $z = \eta(i,j,k,t)$, where H is the depth of the sea floor and η the height of the sea surface. Both H and η are referenced to $z = 0$.

Land - ocean interface : the major flux between continental margins and the ocean is a mass exchange of fresh water through river runoff. Such an exchange modifies the sea surface salinity especially in the vicinity of major river mouths. It can be neglected for short range integrations but has to be taken into account for long term integrations as it influences the characteristics of water masses formed (especially at high latitudes). It is required in order to close the water cycle of the climate system. It is usually specified as a fresh water flux at the air-sea interface in the vicinity of river mouths.

Solid earth - ocean interface : heat and salt fluxes through the sea floor are small, except in special areas of little extent. They are usually neglected in the model [1]. The boundary condition is thus set to no flux of heat and salt across solid boundaries. For momentum, the situation is different. There is no flow across solid boundaries, $i.e.$ the velocity normal to the ocean bottom and coastlines is zero (in other words, the bottom velocity is parallel to solid boundaries). This kinematic boundary condition can be expressed as :

$$w = -\mathbf{U}_h \cdot \nabla_h (H) \tag{2.2}$$

In addition, the ocean exchanges momentum with the earth through frictional processes. Such momentum transfer occurs at small scales in a boundary layer. It must be parameterized in terms of turbulent fluxes using bottom and/or lateral boundary conditions. Its specification depends on the nature of the physical parameterisation used for $\mathbf{D}^{\mathbf{U}}$ in (2.1a). It is discussed in §2.6.1, page 27.

Atmosphere - ocean interface : the kinematic surface condition plus the mass flux of fresh water PE (the precipitation minus evaporation budget) leads to :

$$w = \frac{\partial \eta}{\partial t} + \mathbf{U}_h|_{z=\eta} \cdot \nabla_h (\eta) + P - E \tag{2.3}$$

The dynamic boundary condition, neglecting the surface tension (which removes capillary waves from the system) leads to the continuity of pressure across the interface $z = \eta$. The atmosphere and ocean also exchange horizontal momentum (wind stress), and heat.

Sea ice - ocean interface : the ocean and sea ice exchange heat, salt, fresh water and momentum. The sea surface temperature is constrained to be at the freezing point at the interface. Sea ice salinity is very low ($\sim 4 - 6\,psu$) compared to those of the ocean ($\sim 34\,psu$). The cycle of freezing/melting is associated with fresh water and salt fluxes that cannot be neglected.

[1]In fact, it has been shown that the heat flux associated with the solid Earth cooling ($i.e.$the geothermal heating) is not negligible for the thermohaline circulation of the world ocean (see 4.4.3).

2.2 The Horizontal Pressure Gradient

2.2.1 Pressure Formulation

The total pressure at a given depth z is composed of a surface pressure p_s at a reference geopotential surface ($z = 0$) and a hydrostatic pressure p_h such that : $p(i, j, k, t) = p_s(i, j, t) + p_h(i, j, k, t)$. The latter is computed by integrating (2.1b), assuming that pressure in decibars can be approximated by depth in meters in (2.1f). The hydrostatic pressure is then given by :

$$p_h(i, j, z, t) = \int_{\varsigma=z}^{\varsigma=0} g\ \rho(T, S, \varsigma)\ d\varsigma \tag{2.4}$$

Two strategies can be considered for the surface pressure term : (a) introduce of a new variable η, the free-surface elevation, for which a prognostic equation can be established and solved ; (b) assume that the ocean surface is a rigid lid, on which the pressure (or its horizontal gradient) can be diagnosed. When the former strategy is used, one solution of the free-surface elevation consists of the excitation of external gravity waves. The flow is barotropic and the surface moves up and down with gravity as the restoring force. The phase speed of such waves is high (some hundreds of metres per second) so that the time step would have to be very short if they were present in the model. The latter strategy filters out these waves since the rigid lid approximation implies $\eta = 0$, $i.e.$ the sea surface is the surface $z = 0$. This well known approximation increases the surface wave speed to infinity and modifies certain other longwave dynamics ($e.g.$ barotropic Rossby or planetary waves). In the present release of *NEMO* , both strategies are still available. They are further described in the next two sub-sections.

2.2.2 Free Surface Formulation

In the free surface formulation, a variable η, the sea-surface height, is introduced which describes the shape of the air-sea interface. This variable is solution of a prognostic equation which is established by forming the vertical average of the kinematic surface condition (2.2) :

$$\frac{\partial \eta}{\partial t} = -D + P - E \quad \text{where } D = \nabla \cdot \left[(H + \eta)\ \overline{\mathbf{U}}_h \right] \tag{2.5}$$

and using (2.1b) the surface pressure is given by : $p_s = \rho\, g\, \eta$.

Allowing the air-sea interface to move introduces the external gravity waves (EGWs) as a class of solution of the primitive equations. These waves are barotropic because of hydrostatic assumption, and their phase speed is quite high. Their time scale is short with respect to the other processes described by the primitive equations.

Three choices can be made regarding the implementation of the free surface in the model, depending on the physical processes of interest.

• If one is interested in EGWs, in particular the tides and their interaction with the baroclinic structure of the ocean (internal waves) possibly in shallow seas, then a non linear free surface is the most appropriate. This means that no approximation is made in (2.5) and that the variation of the ocean volume is fully taken into account. Note that in order to study the fast time scales associated with EGWs it is necessary to minimize time filtering effects (use an explicit time scheme with very small time step, or a split-explicit scheme with reasonably small time step, see §5.4.1 or §5.4.1.

• If one is not interested in EGW but rather sees them as high frequency noise, it is possible to apply an explicit filter to slow down the fastest waves while not altering the slow barotropic Rossby waves. If further, an approximative conservation of heat and salt contents is sufficient for the problem solved, then it is sufficient to solve a linearized version of (2.5), which still allows to take into account freshwater fluxes applied at the ocean surface [**?**].

• For process studies not involving external waves nor surface freshwater fluxes, it is possible to use the rigid lid approximation see (next section). The ocean surface is then considered as a fixed surface, so that all external waves are removed from the system.

The filtering of EGWs in models with a free surface is usually a matter of discretisation of the temporal derivatives, using the time splitting method [**??**] or the implicit scheme [**?**]. In *NEMO* , we use a slightly different approach developed by **?** : the damping of EGWs is ensured by introducing an additional force in the momentum equation. (2.1a) becomes :

$$\frac{\partial \mathbf{U}_h}{\partial t} = \mathbf{M} - g\nabla\left(\tilde{\rho}\,\eta\right) - g\,T_c\nabla\left(\widetilde{\rho}\,\partial_t\eta\right) \tag{2.6}$$

where T_c, is a parameter with dimensions of time which characterizes the force, $\widetilde{\rho} = \rho/\rho_o$ is the dimensionless density, and $\mathbf{M}$ represents the collected contributions of the Coriolis, hydrostatic pressure gradient, non-linear and viscous terms in (2.1a).

The new force can be interpreted as a diffusion of vertically integrated volume flux divergence. The time evolution of D is thus governed by a balance of two terms, $-g\;\mathbf{A}\;\eta$ and $g\,T_c\;\;\mathbf{A}\;D$, associated with a propagative regime and a diffusive regime in the temporal spectrum, respectively. In the diffusive regime, the EGWs no longer propagate, $i.e.$ they are stationary and damped. The diffusion regime applies to the modes shorter than T_c. For longer ones, the diffusion term vanishes. Hence, the temporally unresolved EGWs can be damped by choosing $T_c > \Delta t$. **?** demonstrate that (2.6) can be integrated with a leap frog scheme except the additional term which has to be computed implicitly. This is not surprising since the use of a large time step has a necessarily numerical cost. Two gains arise in comparison with the previous formulations. Firstly, the damping of EGWs can be quantified through the magnitude of the additional term. Secondly, the numerical scheme does not need any tuning. Numerical stability is ensured as soon as $T_c > \Delta t$.

When the variations of free surface elevation are small compared to the thickness of the first model layer, the free surface equation (2.5) can be linearized. As emphasized by **?** the linearization of (2.5) has consequences on the conservation of salt in the model. With the nonlinear free surface equation, the time evolution of the total salt content is

$$\frac{\partial}{\partial t}\int\limits_{D\eta} S\;dv = \int\limits_{S} S\;(-\frac{\partial \eta}{\partial t} - D + P - E)\;ds \tag{2.7}$$

where S is the salinity, and the total salt is integrated over the whole ocean volume D_η bounded by the time-dependent free surface. The right hand side (which is an integral over the free surface) vanishes when the nonlinear equation (2.5) is satisfied, so that the salt is perfectly conserved. When the free surface equation is linearized, **?** show that the total salt content integrated in the fixed volume D (bounded by the surface $z = 0$) is no longer conserved :

$$\frac{\partial}{\partial t} \int\limits_D S \; dv = - \int\limits_S S \; \frac{\partial \eta}{\partial t} ds \tag{2.8}$$

The right hand side of (2.8) is small in equilibrium solutions [**?**]. It can be significant when the freshwater forcing is not balanced and the globally averaged free surface is drifting. An increase in sea surface height η results in a decrease of the salinity in the fixed volume D. Even in that case though, the total salt integrated in the variable volume D_η varies much less, since (2.8) can be rewritten as

$$\frac{\partial}{\partial t} \int\limits_{D\eta} S \; dv = \frac{\partial}{\partial t} \left[\int\limits_D S \; dv + \int\limits_S S\eta \; ds \right] = \int\limits_S \eta \; \frac{\partial S}{\partial t} ds \tag{2.9}$$

Although the total salt content is not exactly conserved with the linearized free surface, its variations are driven by correlations of the time variation of surface salinity with the sea surface height, which is a negligible term. This situation contrasts with the case of the rigid lid approximation (following section) in which case freshwater forcing is represented by a virtual salt flux, leading to a spurious source of salt at the ocean surface [**?**].

2.2.3 Rigid-Lid formulation

With the rigid lid approximation, we assume that the ocean surface ($z = 0$) is a rigid lid on which a pressure p_s is exerted. This implies that the vertical velocity at the surface is equal to zero. From the continuity equation (2.1c) and the kinematic condition at the bottom (2.2) (no flux across the bottom), it can be shown that the vertically integrated flow $H\bar{\mathbf{U}}_h$ is non-divergent (where the overbar indicates a vertical average over the whole water column, i.e. from $z = -H$, the ocean bottom, to $z = 0$, the rigid-lid). Thus, $\bar{\mathbf{U}}_\mathrm{h}$ can be derived from a volume transport streamfunction ψ :

$$\overline{\mathbf{U}}_h = \frac{1}{H} \left(\mathbf{k} \times \nabla \psi \right) \tag{2.10}$$

As p_s does not depend on depth, its horizontal gradient is obtained by forming the vertical average of (2.1a) and using (2.10) :

$$\frac{1}{\rho_o} \nabla_h p_s = \overline{\mathbf{M}} - \frac{\partial \overline{\mathbf{U}}_h}{\partial t} = \overline{\mathbf{M}} - \frac{1}{H} \left[\mathbf{k} \times \nabla \left(\frac{\partial \psi}{\partial \mathrm{t}} \right) \right] \tag{2.11}$$

Here $\mathbf{M} = (M_u, M_v)$ represents the collected contributions of the Coriolis, hydrostatic pressure gradient, nonlinear and viscous terms in (2.1a). The time derivative of ψ is the solution of an elliptic equation which is obtained from the vertical component of the curl of (2.11) :

$$\left[\nabla \times \left[\frac{1}{H}\mathbf{k} \times \nabla \left(\frac{\partial \psi}{\partial \mathrm{t}}\right)\right]\right]_z = \left[\nabla \times \overline{\mathbf{M}}\right]_z \tag{2.12}$$

Using the proper boundary conditions, (2.12) can be solved to find $\partial_t\psi$ and thus using (2.11) the horizontal surface pressure gradient. It should be noted that p_s can be computed by taking the divergence of (2.11) and solving the resulting elliptic equation. Thus the surface pressure is a diagnostic quantity that can be recovered for analysis purposes.

A difficulty lies in the determination of the boundary condition on $\partial_t\psi$. The boundary condition on velocity is that there is no flow normal to a solid wall, $i.e.$ the coastlines are streamlines. Therefore (2.12) is solved with the following Dirichlet boundary condition : $\partial_t\psi$ is constant along each coastline of the same continent or of the same island. When all the coastlines are connected (there are no islands), the constant value of $\partial_t\psi$ along the coast can be arbitrarily chosen to be zero. When islands are present in the domain, the value of the barotropic streamfunction will generally be different for each island and for the continent, and will vary with respect to time. So the boundary condition is : $\psi = 0$ along the continent and $\psi = \mu_n$ along island n ($1 \leq n \leq Q$), where Q is the number of islands present in the domain and μ_n is a time dependent variable. A time evolution equation of the unknown μ_n can be found by evaluating the circulation of the time derivative of the vertical average (barotropic) velocity field along a closed contour around each island. Since the circulation of a gradient field along a closed contour is zero, from (2.11) we have :

$$\oint_n \frac{1}{H}\left[\mathbf{k} \times \nabla \left(\frac{\partial \psi}{\partial t}\right)\right] \cdot \mathbf{d}\ell = \oint_n \overline{\mathbf{M}} \cdot \mathbf{d}\ell \qquad 1 \leq n \leq Q \tag{2.13}$$

Since (2.12) is linear, its solution ψ can be decomposed as follows :

$$\psi = \psi_o + \sum_{n=1}^{n=Q} \mu_n \psi_n \tag{2.14}$$

where ψ_o is the solution of (2.12) with $\psi_o = 0$ long all the coastlines, and where ψ_n is the solution of (2.12) with the right-hand side equal to 0, and with $\psi_n = 1$ long the island n, $\psi_n = 0$ along the other boundaries. The function ψ_n is thus independent of time. Introducing (2.14) into (2.13) yields :

$$\left[\oint_n \frac{1}{H}\left[\mathbf{k} \times \nabla\psi_m\right] \cdot \mathbf{d}\ell\right]_{\substack{1\leq m\leqslant Q\\ 1\leq n\leqslant Q}} \left(\frac{\partial \mu_n}{\partial t}\right)_{1\leqslant n\leqslant Q}$$
$$= \left(\oint_n \left[\overline{\mathbf{M}} - \frac{1}{H}\left[\mathbf{k} \times \nabla \left(\frac{\partial \psi_o}{\partial t}\right)\right]\right] \cdot \mathbf{d}\ell\right)_{1\leqslant n\leqslant Q} \tag{2.15}$$

which can be rewritten as :

$$\mathbf{A}\left(\frac{\partial \mu_n}{\partial t}\right)_{1\leqslant n\leqslant Q} = \mathbf{B} \tag{2.16}$$

where **A** is a $Q \times Q$ matrix and **B** is a time dependent vector. As **A** is independent of time, it can be calculated and inverted once. The time derivative of the streamfunction when islands are present is thus given by :

$$\frac{\partial \psi}{\partial t} = \frac{\partial \psi_o}{\partial t} + \sum_{n=1}^{n=Q} \mathbf{A}^{-1} \mathbf{B} \ \psi_n \tag{2.17}$$

2.3 Curvilinear *z*-coordinate System

2.3.1 Tensorial Formalism

In many ocean circulation problems, the flow field has regions of enhanced dynamics ($i.e.$ surface layers, western boundary currents, equatorial currents, or ocean fronts). The representation of such dynamical processes can be improved by specifically increasing the model resolution in these regions. As well, it may be convenient to use a lateral boundary-following coordinate system to better represent coastal dynamics. Moreover, the common geographical coordinate system has a singular point at the North Pole that cannot be easily treated in a global model without filtering. A solution consists of introducing an appropriate coordinate transformation that shifts the singular point onto land [**??**]. As a consequence, it is important to solve the primitive equations in various curvilinear coordinate systems. An efficient way of introducing an appropriate coordinate transform can be found when using a tensorial formalism. This formalism is suited to any multidimensional curvilinear coordinate system. Ocean modellers mainly use three-dimensional orthogonal grids on the sphere (spherical earth approximation), with preservation of the local vertical. Here we give the simplified equations for this particular case. The general case is detailed by **?** in their survey of the conservation laws of fluid dynamics.

Let (**i**,**j**,**k**) be a set of orthogonal curvilinear coordinates on the sphere associated with the positively oriented orthogonal set of unit vectors (**i**,**j**,**k**) linked to the earth such that **k** is the local upward vector and (**i**,**j**) are two vectors orthogonal to **k**, $i.e.$ along geopotential surfaces (Fig.2.3.1). Let (λ, φ, z) be the geographical coordinate system in which a position is defined by the latitude $\varphi(i,j)$, the longitude $\lambda(i,j)$ and the distance from the centre of the earth $a+z(k)$ where a is the earth's radius and z the altitude above a reference sea level (Fig.2.3.1). The local deformation of the curvilinear coordinate system is given by e_1, e_2 and e_3, the three scale factors :

$$\begin{aligned} e_1 &= (a+z) \ \left[\left(\frac{\partial \lambda}{\partial i} \cos \varphi \right)^2 + \left(\frac{\partial \varphi}{\partial i} \right)^2 \right]^{1/2} \\ e_2 &= (a+z) \ \left[\left(\frac{\partial \lambda}{\partial j} \cos \varphi \right)^2 + \left(\frac{\partial \varphi}{\partial j} \right)^2 \right]^{1/2} \\ e_3 &= \left(\frac{\partial z}{\partial k} \right) \end{aligned} \tag{2.18}$$

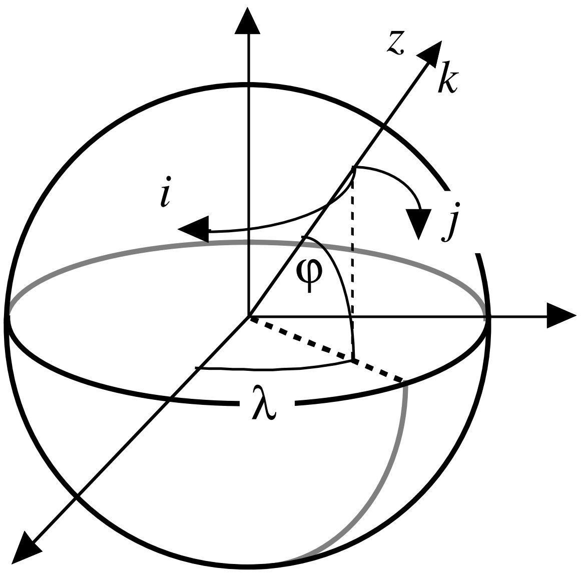


FIG. 2.2 – the geographical coordinate system (λ, φ, z) and the curvilinear coordinate system (**i**,**j**,**k**).

Since the ocean depth is far smaller than the earth's radius, $a + z$, can be replaced by a in (2.18) (thin-shell approximation). The resulting horizontal scale factors e_1, e_2 are independent of k while the vertical scale factor is a single function of k as **k** is parallel to **z**. The scalar and vector operators that appear in the primitive equations (Eqs. (2.1a) to (2.1f)) can be written in the tensorial form, invariant in any orthogonal horizontal curvilinear coordinate system transformation :

$$\nabla q = \frac{1}{e_1}\frac{\partial q}{\partial i}\,\mathbf{i} + \frac{1}{e_2}\frac{\partial q}{\partial j}\,\mathbf{j} + \frac{1}{e_3}\frac{\partial q}{\partial k}\,\mathbf{k} \tag{2.19a}$$

$$\nabla \cdot \mathbf{A} = \frac{1}{e_1\,e_2}\left[\frac{\partial\,(e_2\,a_1)}{\partial i} + \frac{\partial\,(e_1\,a_2)}{\partial j}\right] + \frac{1}{e_3}\left[\frac{\partial a_3}{\partial k}\right] \tag{2.19b}$$

$$\begin{aligned} \nabla \times \mathbf{A} = \left[\frac{1}{\mathrm{e}_2}\frac{\partial \mathrm{a}_3}{\partial \mathrm{j}} - \frac{1}{\mathrm{e}_3}\frac{\partial \mathrm{a}_2}{\partial \mathrm{k}}\right]\mathbf{i} + \left[\frac{1}{e_3}\frac{\partial a_1}{\partial k} - \frac{1}{e_1}\frac{\partial a_3}{\partial i}\right]\mathbf{j} \\ + \frac{1}{e_1 e_2}\left[\frac{\partial\,(e_2 a_2)}{\partial i} - \frac{\partial\,(e_1 a_1)}{\partial j}\right]\mathbf{k} \end{aligned} \tag{2.19c}$$

$$\Delta q = \nabla \cdot (\nabla q) \tag{2.19d}$$

$$\Delta \mathbf{A} = \nabla\,(\nabla \cdot \mathbf{A}) - \nabla \times (\nabla \times \mathbf{A}) \tag{2.19e}$$

where q is a scalar quantity and $\mathbf{A} = (a_1, a_2, a_3)$ a vector in the (i, j, k) coordinate system.

2.3.2 Continuous Model Equations

In order to express the Primitive Equations in tensorial formalism, it is necessary to compute the horizontal component of the non-linear and viscous terms of the equation using (2.19a)) to (2.19e). Let us set **U** $= (\mathrm{u}, \mathrm{v}, \mathrm{w}) = \mathbf{U}_\mathrm{h} + \mathrm{w}\ \mathbf{k}$, the velocity in the (i, j, k) coordinate system and define the relative vorticity ζ and the divergence of the horizontal velocity field χ, by :

$$\zeta = \frac{1}{e_1 e_2}\left[\frac{\partial\,(e_2\,v)}{\partial i} - \frac{\partial\,(e_1\,u)}{\partial j}\right] \tag{2.20}$$

$$\chi = \frac{1}{e_1 e_2}\left[\frac{\partial\,(e_2\,u)}{\partial i} + \frac{\partial\,(e_1\,v)}{\partial j}\right] \tag{2.21}$$

Using the fact that the horizontal scale factors e_1 and e_2 are independent of k and that e_3 is a function of the single variable k, the nonlinear term of (2.1a) can be transformed as follows :

$$\left[(\nabla\times\mathbf{U})\times\mathbf{U} + \frac{1}{2}\nabla\left(\mathbf{U}^2\right)\right]_h$$

$$= \begin{pmatrix} \left[\frac{1}{e_3}\frac{\partial u}{\partial k} - \frac{1}{e_1}\frac{\partial w}{\partial i}\right] w - \zeta\ v \\ \zeta\ u - \left[\frac{1}{e_2}\frac{\partial w}{\partial j} - \frac{1}{e_3}\frac{\partial v}{\partial k}\right]\ w \end{pmatrix} + \frac{1}{2}\begin{pmatrix} \frac{1}{e_1}\frac{\partial\left(u^2+v^2+w^2\right)}{\partial i} \\ \frac{1}{e_2}\frac{\partial\left(u^2+v^2+w^2\right)}{\partial j} \end{pmatrix}$$

$$= \begin{pmatrix} -\zeta\ v \\ \zeta\ u \end{pmatrix} + \frac{1}{2}\begin{pmatrix} \frac{1}{e_1}\frac{\partial\left(u^2+v^2\right)}{\partial i} \\ \frac{1}{e_2}\frac{\partial\left(u^2+v^2\right)}{\partial j} \end{pmatrix} + \frac{1}{e_3}\begin{pmatrix} w\ \frac{\partial u}{\partial k} \\ w\ \frac{\partial v}{\partial k} \end{pmatrix} - \begin{pmatrix} \frac{w}{e_1}\frac{\partial w}{\partial i} - \frac{1}{2e_1}\frac{\partial w^2}{\partial i} \\ \frac{w}{e_2}\frac{\partial w}{\partial j} - \frac{1}{2e_2}\frac{\partial w^2}{\partial j} \end{pmatrix}$$

The last term of the right hand side is obviously zero, and thus the nonlinear term of (2.1a) is written in the (i, j, k) coordinate system :

$$\left[(\nabla\times\mathbf{U})\times\mathbf{U} + \frac{1}{2}\nabla\left(\mathbf{U}^2\right)\right]_h = \zeta\ \mathbf{k}\times\mathbf{U}_h + \frac{1}{2}\nabla_h\left(\mathbf{U}_h^2\right) + \frac{1}{e_3}w\frac{\partial\mathbf{U}_h}{\partial k} \tag{2.22}$$

This is the so-called *vector invariant form* of the momentum advection term. For some purposes, it can be advantageous to write this term in the so-called flux form, *i.e.* to write it as the divergence of fluxes. For example, the first component of (2.22) (the i-component) is transformed as follows :

$$\left[(\nabla\times\mathbf{U})\times\mathbf{U} + \tfrac{1}{2}\nabla\left(\mathbf{U}^2\right)\right]_i = -\zeta\ v + \tfrac{1}{2\,e_1}\tfrac{\partial\left(u^2+v^2\right)}{\partial i} + \tfrac{1}{e_3}w\ \tfrac{\partial u}{\partial k}$$

$$= \tfrac{1}{e_1\,e_2}\left(-v\tfrac{\partial(e_2\,v)}{\partial i} + v\tfrac{\partial(e_1\,u)}{\partial j}\right) + \tfrac{1}{e_1 e_2}\left(+e_2\ u\tfrac{\partial u}{\partial i} + e_2\ v\tfrac{\partial v}{\partial i}\right) + \tfrac{1}{e_3}\left(w\ \tfrac{\partial u}{\partial k}\right)$$

$$\begin{aligned} = \tfrac{1}{e_1\,e_2}\Big\{ &-\left(v^2\tfrac{\partial e_2}{\partial i} + e_2\,v\tfrac{\partial v}{\partial i}\right) + \left(\tfrac{\partial(e_1\,u\,v)}{\partial j} - e_1\,u\tfrac{\partial v}{\partial j}\right) \\ &+ \left(\tfrac{\partial(e_2 u\,u)}{\partial i} - u\tfrac{\partial(e_2 u)}{\partial i}\right) + e_2 v\tfrac{\partial v}{\partial i}\Big\} + \tfrac{1}{e_3}\left(\tfrac{\partial(w\,u)}{\partial k} - u\tfrac{\partial w}{\partial k}\right) \end{aligned}$$

$$= \frac{1}{e_1\, e_2}\left(\frac{\partial(e_2\, u\, u)}{\partial i}+\frac{\partial(e_1\, u\, v)}{\partial j}\right)+\frac{1}{e_3}\frac{\partial(w\, u)}{\partial k}$$
$$+\frac{1}{e_1 e_2}\left(-u\left(\frac{\partial(e_1 v)}{\partial j}-v\,\frac{\partial e_1}{\partial j}\right)-u\frac{\partial(e_2 u)}{\partial i}\right)-\frac{1}{e_3}\frac{\partial w}{\partial k}u+\frac{1}{e_1 e_2}\left(-v^2\frac{\partial e_2}{\partial i}\right)$$

$$= \nabla\cdot(\mathbf{U}\, u)-(\nabla\cdot\mathbf{U})\; u+\frac{1}{e_1 e_2}\left(-v^2\frac{\partial e_2}{\partial i}+uv\,\frac{\partial e_1}{\partial j}\right)$$

as $\nabla\cdot\mathbf{U}\;=0$ (incompressibility) it comes :

$$= \nabla\cdot(\mathbf{U}\, u)+\frac{1}{e_1 e_2}\left(v\,\frac{\partial e_2}{\partial i}-u\,\frac{\partial e_1}{\partial j}\right)(-v)$$

The flux form of the momentum advection term is therefore given by :

$$\left[(\nabla\times\mathbf{U})\times\mathbf{U}+\frac{1}{2}\nabla\left(\mathbf{U}^2\right)\right]_h$$
$$= \nabla\cdot\begin{pmatrix}\mathbf{U}\, u\\ \mathbf{U}\, v\end{pmatrix}+\frac{1}{e_1 e_2}\left(v\frac{\partial e_2}{\partial i}-u\frac{\partial e_1}{\partial j}\right)\mathbf{k}\times\mathbf{U}_h \quad (2.23)$$

The flux form has two terms, the first one is expressed as the divergence of momentum fluxes (hence the flux form name given to this formulation) and the second one is due to the curvilinear nature of the coordinate system used. The latter is called the *metric* term and can be viewed as a modification of the Coriolis parameter :

$$f\to f+\frac{1}{e_1\; e_2}\left(v\frac{\partial e_2}{\partial i}-u\frac{\partial e_1}{\partial j}\right) \quad (2.24)$$

Note that in the case of geographical coordinate, $i.e.$ when $(i,j)\;\to\;(\lambda,\varphi)$ and $(e_1,e_2)\to(a\;\cos\varphi,a)$, we recover the commonly used modification of the Coriolis parameter $f\to f+(u/a)\tan\varphi$.

To sum up, the equations solved by the ocean model can be written in the following tensorial formalism :

• *momentum equations* :
vector invariant form :

$$\frac{\partial u}{\partial t}=+\left(\zeta+f\right)\,v-\frac{1}{2\,e_1}\frac{\partial}{\partial i}\left(u^2+v^2\right)-\frac{1}{e_3}w\frac{\partial u}{\partial k}$$
$$-\frac{1}{e_1}\frac{\partial}{\partial i}\left(\frac{p_s+p_h}{\rho_o}\right)+D_u^{\mathbf{U}}+F_u^{\mathbf{U}} \quad (2.25a)$$

$$\frac{\partial v}{\partial t}=-\left(\zeta+f\right)\,u-\frac{1}{2\,e_2}\frac{\partial}{\partial j}\left(u^2+v^2\right)-\frac{1}{e_3}w\frac{\partial v}{\partial k}$$
$$-\frac{1}{e_2}\frac{\partial}{\partial j}\left(\frac{p_s+p_h}{\rho_o}\right)+D_v^{\mathbf{U}}+F_v^{\mathbf{U}} \quad (2.25b)$$

flux form :

$$\begin{aligned}\frac{\partial u}{\partial t} = &+ \left(f + \frac{1}{e_1\, e_2}\left(v\frac{\partial e_2}{\partial i} - u\frac{\partial e_1}{\partial j}\right)\right)\, v \\ &- \frac{1}{e_1\, e_2}\left(\frac{\partial\left(e_2\, u\, u\right)}{\partial i} + \frac{\partial\left(e_1\, v\, u\right)}{\partial j}\right) - \frac{1}{e_3}\frac{\partial\left(w\, u\right)}{\partial k} \\ &- \frac{1}{e_1}\frac{\partial}{\partial i}\left(\frac{p_s + p_h}{\rho_o}\right) + D_u^{\mathbf{U}} + F_u^{\mathbf{U}}\end{aligned} \tag{2.26a}$$

$$\begin{aligned}\frac{\partial v}{\partial t} = &- \left(f + \frac{1}{e_1\, e_2}\left(v\frac{\partial e_2}{\partial i} - u\frac{\partial e_1}{\partial j}\right)\right)\, u \\ &\frac{1}{e_1\, e_2}\left(\frac{\partial\left(e_2\, u\, v\right)}{\partial i} + \frac{\partial\left(e_1\, v\, v\right)}{\partial j}\right) - \frac{1}{e_3}\frac{\partial\left(w\, v\right)}{\partial k} \\ &- \frac{1}{e_2}\frac{\partial}{\partial j}\left(\frac{p_s + p_h}{\rho_o}\right) + D_v^{\mathbf{U}} + F_v^{\mathbf{U}}\end{aligned} \tag{2.26b}$$

where ζ is given by (2.20) and the surface pressure gradient formulation depends on the one of the free surface :

$*$ free surface formulation

$$\frac{1}{\rho_o}\nabla_h p_s = \begin{pmatrix} \frac{g}{e_1}\frac{\partial \eta}{\partial i} \\ \frac{g}{e_2}\frac{\partial \eta}{\partial j} \end{pmatrix} \qquad \text{where } \eta \text{ is solution of (2.5)} \tag{2.27}$$

$*$ rigid-lid approximation

$$\frac{1}{\rho_o}\nabla_h p_s = \begin{pmatrix} \overline{M}_u + \frac{1}{H\, e_2}\frac{\partial}{\partial j}\left(\frac{\partial \psi}{\partial t}\right) \\ \overline{M}_v - \frac{1}{H\, e_1}\frac{\partial}{\partial i}\left(\frac{\partial \psi}{\partial t}\right) \end{pmatrix} \tag{2.28}$$

where $\mathbf{M} = (M_u, M_v)$ represents the collected contributions of nonlinear, viscosity and hydrostatic pressure gradient terms in (2.25) and the overbar indicates a vertical average over the whole water column ($i.e.$ from $z = -H$, the ocean bottom, to $z = 0$, the rigid-lid), and where the time derivative of ψ is the solution of an elliptic equation :

$$\begin{aligned}\frac{\partial}{\partial i}\left[\frac{e_2}{H\, e_1}\frac{\partial}{\partial i}\left(\frac{\partial \psi}{\partial t}\right)\right] + \frac{\partial}{\partial j}\left[\frac{e_1}{H\, e_2}\frac{\partial}{\partial j}\left(\frac{\partial \psi}{\partial t}\right)\right] = \\ \frac{\partial}{\partial i}\left(e_2 \overline{M}_v\right) - \frac{\partial}{\partial j}\left(e_1 \overline{M}_u\right)\end{aligned} \tag{2.29}$$

The vertical velocity and the hydrostatic pressure are diagnosed from the following equations :

$$\frac{\partial w}{\partial k} = -\chi\, e_3 \tag{2.30}$$

$$\frac{\partial p_h}{\partial k} = -\rho\, g\, e_3 \tag{2.31}$$

where the divergence of the horizontal velocity, χ is given by (2.21).

• *tracer equations* :

$$\frac{\partial T}{\partial t} = -\frac{1}{e_1 e_2}\left[\frac{\partial \left(e_2 T\, u\right)}{\partial i} + \frac{\partial \left(e_1 T\, v\right)}{\partial j}\right] - \frac{1}{e_3}\frac{\partial \left(T\, w\right)}{\partial k} + D^T + F^T \tag{2.32}$$

$$\frac{\partial S}{\partial t} = -\frac{1}{e_1 e_2}\left[\frac{\partial \left(e_2 S\, u\right)}{\partial i} + \frac{\partial \left(e_1 S\, v\right)}{\partial j}\right] - \frac{1}{e_3}\frac{\partial \left(S\, w\right)}{\partial k} + D^S + F^S \tag{2.33}$$

$$\rho = \rho\left(T, S, z(k)\right) \tag{2.34}$$

The expression of $\mathbf{D}^U$, D^S and D^T depends on the subgrid scale parameterisation used. It will be defined in §2.6.1. The nature and formulation of $\mathbf{F^U}$, F^T and F^S, the surface forcing terms, are discussed in Chapter 6.

2.4 Curvilinear *z**-coordinate System

In that case, the free surface equation is nonlinear, and the variations of volume are fully taken into account. These coordinates systems is presented in a report [**?**] available on the *NEMO* web site.

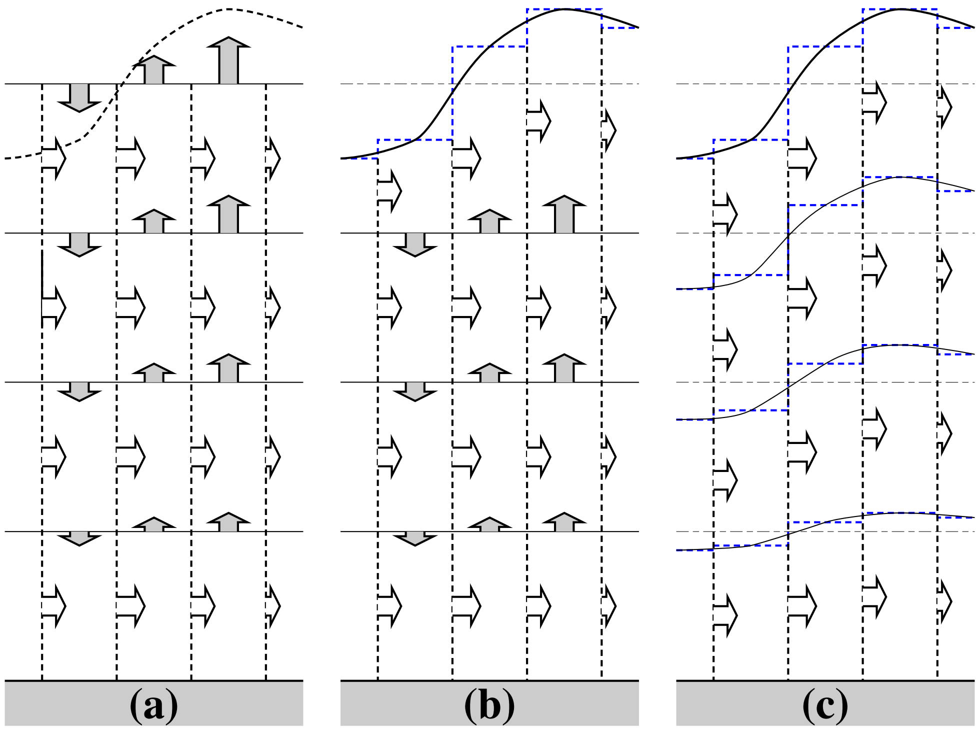


FIG. 2.3 – (a) z-coordinate in linear free-surface case ; (b) $z-$coordinate in non-linear free surface case (c) re-scaled height coordinate (become popular as the *z**-coordinate [**?**]).

2.5 Curvilinear *s*-coordinate System

2.5.1 Introduction

Several important aspects of the ocean circulation are influenced by bottom topography. Of course, the most important is that bottom topography determines deep ocean sub-basins, barriers, sills and channels that strongly constrain the path of water masses, but more subtle effects exist. For example, the topographic β-effect is usually larger than the planetary one along continental slopes. Topographic Rossby waves can be excited and can interact with the mean current. In the $z-$coordinate system presented in the previous section (§2.3), $z-$surfaces are geopotential surfaces. The bottom topography is discretised by steps. This often leads to a misrepresentation of a gradually sloping bottom and to large localized depth gradients associated with large localized vertical velocities. The response to such a velocity field often leads to numerical dispersion effects. One solution to strongly reduce this error is to use a partial step representation of bottom topography instead of a full step one **?**. Another solution is to introduce a terrain-following coordinate system (hereafter $s-$coordinate)

The s-coordinate avoids the discretisation error in the depth field since the layers of computation are gradually adjusted with depth to the ocean bottom. Relatively small topographic features as well as gentle, large-scale slopes of the sea floor in the deep ocean, which would be ignored in typical z-model applications with the largest grid spacing at greatest depths, can easily be represented (with relatively low vertical resolution). A terrain-following model (hereafter $s-$model) also facilitates the modelling of the boundary layer flows over a large depth range, which in the framework of the z-model would require high vertical resolution over the whole depth range. Moreover, with a s-coordinate it is possible, at least in principle, to have the bottom and the sea surface as the only boundaries of the domain (nomore lateral boundary condition to specify). Nevertheless, a s-coordinate also has its drawbacks. Perfectly adapted to a homogeneous ocean, it has strong limitations as soon as stratification is introduced. The main two problems come from the truncation error in the horizontal pressure gradient and a possibly increased diapycnal diffusion. The horizontal pressure force in s-coordinate consists of two terms (see Appendix A),

$$\nabla p|_z = \nabla p|_s - \frac{\partial p}{\partial s} \nabla z|_s \tag{2.35}$$

The second term in (2.35) depends on the tilt of the coordinate surface and introduces a truncation error that is not present in a z-model. In the special case of a $\sigma-$coordinate (i.e. a depth-normalised coordinate system $\sigma = z/H$), **?** and **?** have given estimates of the magnitude of this truncation error. It depends on topographic slope, stratification, horizontal and vertical resolution, the equation of state, and the finite difference scheme. This error limits the possible topographic slopes that a model can handle at a given horizontal and vertical resolution. This is a severe restriction for large-scale applications using realistic bottom topography. The large-scale slopes require high horizontal resolution, and the computational cost becomes prohibitive. This problem can be at least partially overcome by mixing s-coordinate and step-like representation of bottom topography [**???**]. However, the definition of the model domain vertical coordinate becomes then a non-trivial thing for a realistic bottom topography : a envelope topography is defined in s-coordinate on which a full or partial step bottom topography is then applied in order to adjust the model depth to the observed one (see §3.3.

For numerical reasons a minimum of diffusion is required along the coordinate surfaces of any finite difference model. It causes spurious diapycnal mixing when coordinate surfaces do not coincide with isoneutral surfaces. This is the case for a z-model as well as for a s-model. However, density varies more strongly on $s-$surfaces than on horizontal surfaces in regions of large topographic slopes, implying larger diapycnal diffusion in a s-model than in a z-model. Whereas such a diapycnal diffusion in a z-model tends to weaken horizontal density (pressure) gradients and thus the horizontal circulation, it usually reinforces these gradients in a s-model, creating spurious circulation. For example, imagine an isolated bump of topography in an ocean at rest with a horizontally uniform stratification. Spurious diffusion along s-surfaces will induce a bump of isoneutral surfaces over the topography, and thus will generate there a baroclinic eddy. In contrast, the ocean will stay at rest in a z-model. As for the truncation error, the problem can be reduced by introducing the terrain-following coordinate below the strongly stratified portion of the water column ($i.e.$ the main thermocline) [**?**]. An alternate solution consists of rotating the lateral diffusive tensor to geopotential or to isoneutral surfaces (see §2.6.2. Unfortunately, the slope of isoneutral surfaces relative to the s-surfaces can very large, strongly exceeding the stability limit of such a operator when it is discretized (see Chapter 8).

The $s-$coordinates introduced here [**??**] differ mainly in two aspects from similar models : it allows a representation of bottom topography with mixed full or partial step-like/terrain following topography ; It also offers a completely general transformation, $s = s(i, j, z)$ for the vertical coordinate.

2.5.2 The *s*-coordinate Formulation

Starting from the set of equations established in §2.3 for the special case $k = z$ and thus $e_3 = 1$, we introduce an arbitrary vertical coordinate $s = s(i, j, k, t)$, which includes z-, z*- and $\sigma-$coordinates as special cases ($s = z$, $s =$ z*, and $s = \sigma = z/H$ or $= z/\left(H + \eta\right)$, resp.). A formal derivation of the transformed equations is given in Appendix A. Let us define the vertical scale factor by $e_3 = \partial_s z$ (e_3 is now a function of (i, j, k, t)), and the slopes in the (**i**,**j**) directions between $s-$ and $z-$surfaces by :

$$\sigma_1 = \frac{1}{e_1}\ \left.\frac{\partial z}{\partial i}\right|_s \quad \text{, and} \quad \sigma_2 = \frac{1}{e_2}\ \left.\frac{\partial z}{\partial j}\right|_s \tag{2.36}$$

We also introduce ω, a dia-surface velocity component, defined as the velocity relative to the moving s-surfaces and normal to them :

$$\omega = w - e_3\,\frac{\partial s}{\partial t} - \sigma_1\,u - \sigma_2\,v \tag{2.37}$$

The equations solved by the ocean model (2.1) in $s-$coordinate can be written as follows :

* momentum equation :

$$\begin{aligned}\frac{1}{e_3}\frac{\partial\left(e_3\,u\right)}{\partial t}=&+\left(\zeta+f\right)\,v-\frac{1}{2\,e_1}\frac{\partial}{\partial i}\left(u^2+v^2\right)-\frac{1}{e_3}\omega\frac{\partial u}{\partial k}\\&-\frac{1}{e_1}\frac{\partial}{\partial i}\left(\frac{p_s+p_h}{\rho_o}\right)+g\frac{\rho}{\rho_o}\sigma_1+D_u^{\mathbf{U}}+F_u^{\mathbf{U}}\end{aligned}\tag{2.38}$$

$$\begin{aligned}\frac{1}{e_3}\frac{\partial\left(e_3\,v\right)}{\partial t}=&-\left(\zeta+f\right)\,u-\frac{1}{2\,e_2}\frac{\partial}{\partial j}\left(u^2+v^2\right)-\frac{1}{e_3}\omega\frac{\partial v}{\partial k}\\&-\frac{1}{e_2}\frac{\partial}{\partial j}\left(\frac{p_s+p_h}{\rho_o}\right)+g\frac{\rho}{\rho_o}\sigma_2+D_v^{\mathbf{U}}+F_v^{\mathbf{U}}\end{aligned}\tag{2.39}$$

where the relative vorticity, ζ, the surface pressure gradient, and the hydrostatic pressure have the same expressions as in z-coordinates although they do not represent exactly the same quantities. ω is provided by the continuity equation (see Appendix A) :

$$\frac{\partial e_3}{\partial t}+e_3\,\chi+\frac{\partial\omega}{\partial s}=0\qquad\text{with}\quad\chi=\frac{1}{e_1e_2e_3}\left[\frac{\partial\left(e_2e_3\,u\right)}{\partial i}+\frac{\partial\left(e_1e_3\,v\right)}{\partial j}\right]\tag{2.40}$$

* tracer equations :

$$\begin{aligned}\frac{1}{e_3}\frac{\partial\left(e_3\,T\right)}{\partial t}=&-\frac{1}{e_1e_2e_3}\left[\frac{\partial\left(e_2e_3\,u\,T\right)}{\partial i}+\frac{\partial\left(e_1e_3\,v\,T\right)}{\partial j}\right]\\&-\frac{1}{e_3}\frac{\partial\left(T\,\omega\right)}{\partial k}+D^T+F^S\end{aligned}\tag{2.41}$$

$$\begin{aligned}\frac{1}{e_3}\frac{\partial\left(e_3\,S\right)}{\partial t}=&-\frac{1}{e_1e_2e_3}\left[\frac{\partial\left(e_2e_3\,u\,S\right)}{\partial i}+\frac{\partial\left(e_1e_3\,v\,S\right)}{\partial j}\right]\\&-\frac{1}{e_3}\frac{\partial\left(S\,\omega\right)}{\partial k}+D^S+F^S\end{aligned}\tag{2.42}$$

The equation of state has the same expression as in z-coordinate, and similar expressions are used for mixing and forcing terms.

2.6 Subgrid Scale Physics

The primitive equations describe the behaviour of a geophysical fluid at space and time scales larger than a few kilometres in the horizontal, a few meters in the vertical and a few minutes. They are usually solved at larger scales : the specified grid spacing and time step of the numerical model. The effects of smaller scale motions (coming from the advective terms in the Navier-Stokes equations) must be represented entirely in terms of large-scale patterns to close the equations. These effects appear in the equations as the divergence of turbulent fluxes ($i.e.$ fluxes associated with the mean correlation of small scale perturbations). Assuming a turbulent closure hypothesis is equivalent to choose a formulation for these fluxes. It is usually called the subgrid scale physics. It must be emphasized that this is the weakest part of the primitive equations, but also one of the most important for long-term simulations as small scale processes *in fine* balance the surface input of kinetic energy and heat.

The control exerted by gravity on the flow induces a strong anisotropy between the lateral and vertical motions. Therefore subgrid-scale physics $\mathbf{D}^{\mathbf{U}}$, D^S and D^T in (2.1a), (2.1d) and (2.1e) are divided into a lateral part $\mathbf{D}^{l\mathbf{U}}$, D^{lS} and D^{lT} and a vertical part $\mathbf{D}^{vU}$, D^{vS} and D^{vT}. The formulation of these terms and their underlying physics are briefly discussed in the next two subsections.

2.6.1 Vertical Subgrid Scale Physics

The model resolution is always larger than the scale at which the major sources of vertical turbulence occur (shear instability, internal wave breaking...). Turbulent motions are thus never explicitly solved, even partially, but always parameterized. The vertical turbulent fluxes are assumed to depend linearly on the gradients of large-scale quantities (for example, the turbulent heat flux is given by $\overline{T'w'} = -A^{vT}\partial_z\overline{T}$, where A^{vT} is an eddy coefficient). This formulation is analogous to that of molecular diffusion and dissipation. This is quite clearly a necessary compromise : considering only the molecular viscosity acting on large scale severely underestimates the role of turbulent diffusion and dissipation, while an accurate consideration of the details of turbulent motions is simply impractical. The resulting vertical momentum and tracer diffusive operators are of second order :

$$\begin{aligned}
&\mathbf{D}^{v\mathbf{U}} = \frac{\partial}{\partial z}\left(A^{vm}\frac{\partial \mathbf{U}_h}{\partial z}\right) \ , \\
&D^{vT} = \frac{\partial}{\partial z}\left(A^{vT}\frac{\partial T}{\partial z}\right) \ , \quad D^{vS} = \frac{\partial}{\partial z}\left(A^{vT}\frac{\partial S}{\partial z}\right)
\end{aligned} \tag{2.43}$$

where A^{vm} and A^{vT} are the vertical eddy viscosity and diffusivity coefficients, respectively. At the sea surface and at the bottom, turbulent fluxes of momentum, heat and salt must be specified (see Chap. 6 and 9 and §4.5). All the vertical physics is embedded in the specification of the eddy coefficients. They can be assumed to be either constant, or function of the local fluid properties ($e.g.$ Richardson number, Brunt-Vaisälä frequency...), or computed from a turbulent closure model. The choices available in *NEMO* are discussed in §9).

2.6.2 Lateral Diffusive and Viscous Operators Formulation

Lateral turbulence can be roughly divided into a mesoscale turbulence associated with eddies (which can be solved explicitly if the resolution is sufficient since their underlying physics are included in the primitive equations), and a sub mesoscale turbulence which is never explicitly solved even partially, but always parameterized. The formulation of lateral eddy fluxes depends on whether the mesoscale is below or above the grid-spacing ($i.e.$ the model is eddy-resolving or not).

In non-eddy-resolving configurations, the closure is similar to that used for the vertical physics. The lateral turbulent fluxes are assumed to depend linearly on the lateral gradients of large-scale quantities. The resulting lateral diffusive and dissipative operators are of second order. Observations show that lateral mixing induced by mesoscale turbulence tends to be along isopycnal surfaces (or more precisely neutral surfaces **?**) rather than across them. As the slope of neutral surfaces is small in the ocean, a common approximation is to assume that the 'lateral' direction is the horizontal, $i.e.$ the lateral mixing is performed along geopotential surfaces. This leads to a geopotential second order operator for lateral subgrid scale physics. This assumption can be relaxed : the eddy-induced turbulent fluxes can be better approached by assuming that they depend linearly on the gradients of large-scale quantities computed along neutral surfaces. In such a case, the diffusive operator is an isoneutral second order operator and it has components in the three space directions. However, both horizontal and isoneutral operators have no effect on mean ($i.e.$ large scale) potential energy whereas potential energy is a main source of turbulence (through baroclinic instabilities). **?** have proposed a parameterisation of mesoscale eddy-induced turbulence which associates an eddy-induced velocity to the isoneutral diffusion. Its mean effect is to reduce the mean potential energy of the ocean. This leads to a formulation of lateral subgrid-scale physics made up of an isoneutral second order operator and an eddy induced advective part. In all these lateral diffusive formulations, the specification of the lateral eddy coefficients remains the problematic point as there is no really satisfactory formulation of these coefficients as a function of large-scale features.

In eddy-resolving configurations, a second order operator can be used, but usually a more scale selective one (biharmonic operator) is preferred as the grid-spacing is usually not small enough compared to the scale of the eddies. The role devoted to the subgrid-scale physics is to dissipate the energy that cascades toward the grid scale and thus ensures the stability of the model while not interfering with the solved mesoscale activity. Another approach is becoming more and more popular : instead of specifying explicitly a sub-grid scale term in the momentum and tracer time evolution equations, one uses a advective scheme which is diffusive enough to maintain the model stability. It must be emphasised that then, all the sub-grid scale physics is in this case include in the formulation of the advection scheme.

All these parameterisations of subgrid scale physics present advantages and drawbacks. There are not all available in *NEMO* . In the z-coordinate formulation, five options are offered for active tracers (temperature and salinity) : second order geopotential operator, second order isoneutral operator, **?** parameterisation, fourth order geopotential operator, and various slightly diffusive advection schemes. The same options are available for momentum, except **?** parameterisation which only involves tracers. In the s-coordinate formulation, additional options are offered for tracers : second order operator acting along $s-$surfaces, and for momentum : fourth order operator acting along $s-$surfaces (see §8).

lateral second order tracer diffusive operator

The lateral second order tracer diffusive operator is defined by (see Appendix B) :

$$D^{lT} = \nabla . \left(A^{lT} \; \Re \; \nabla T \right) \qquad \text{with} \quad \Re = \begin{pmatrix} 1 & 0 & -r_1 \\ 0 & 1 & -r_2 \\ -r_1 & -r_2 & r_1^2 + r_2^2 \end{pmatrix} \tag{2.44}$$

where r_1 and r_2 are the slopes between the surface along which the diffusive operator acts and the model level ($e.g.$ z- or s-surfaces). Note that the formulation (2.44) is exact for the rotation between geopotential and s-surfaces, while it is only an approximation for the rotation between isoneutral and z- or s-surfaces. Indeed, in the latter case, two assumptions are made to simplify (2.44) [**?**]. First, the horizontal contribution of the dianeutral mixing is neglected since the ratio between iso and dia-neutral diffusive coefficients is known to be several orders of magnitude smaller than unity. Second, the two isoneutral directions of diffusion are assumed to be independent since the slopes are generally less than 10^{-2} in the ocean (see Appendix B).

For *geopotential* diffusion, r_1 and r_2 are the slopes between the geopotential and computational surfaces : in z-coordinates they are zero ($r_1 = r_2 = 0$) while in s-coordinate (including *z** case) they are equal to σ_1 and σ_2, respectively (see (2.36)).

For *isoneutral* diffusion r_1 and r_2 are the slopes between the isoneutral and computational surfaces. Therefore, they have a same expression in z- and s-coordinates :

$$r_1 = \frac{e_3}{e_1} \left(\frac{\partial \rho}{\partial i} \right) \left(\frac{\partial \rho}{\partial k} \right)^{-1} \quad , \quad r_1 = \frac{e_3}{e_1} \left(\frac{\partial \rho}{\partial i} \right) \left(\frac{\partial \rho}{\partial k} \right)^{-1} \tag{2.45}$$

When the *Eddy Induced Velocity* parametrisation (eiv) [**?**] is used, an additional tracer advection is introduced in combination with the isoneutral diffusion of tracers :

$$D^{lT} = \nabla \cdot \left(A^{lT} \; \Re \; \nabla T \right) + \nabla \cdot \left(\mathbf{U}^* \, T \right) \tag{2.46}$$

where $\mathbf{U}^* = (u^*, v^*, w^*)$ is a non-divergent, eddy-induced transport velocity. This velocity field is defined by :

$$\begin{aligned} u^* &= +\frac{1}{e_3} \frac{\partial}{\partial k} \left[A^{eiv} \; \tilde{r}_1 \right] \\ v^* &= +\frac{1}{e_3} \frac{\partial}{\partial k} \left[A^{eiv} \; \tilde{r}_2 \right] \\ w^* &= -\frac{1}{e_1 e_2} \left[\frac{\partial}{\partial i} \left(A^{eiv} \; e_2 \, \tilde{r}_1 \right) + \frac{\partial}{\partial j} \left(A^{eiv} \; e_1 \, \tilde{r}_2 \right) \right] \end{aligned} \tag{2.47}$$

where A^{eiv} is the eddy induced velocity coefficient (or equivalently the isoneutral thickness diffusivity coefficient), and $\tilde{r}_1$ and $\tilde{r}_2$ are the slopes between isoneutral and *geopotential* surfaces and thus depends on the coordinate considered :

$$\tilde{r}_n = \begin{cases} r_n & \text{in } z\text{-coordinate} \\ r_n + \sigma_n & \text{in z* and } s\text{-coordinates} \end{cases} \qquad \text{where } n = 1, 2 \tag{2.48}$$

The normal component of the eddy induced velocity is zero at all the boundaries. This can be achieved in a model by tapering either the eddy coefficient or the slopes to zero in the vicinity of the boundaries. The latter strategy is used in *NEMO* (cf. Chap. 8).

lateral fourth order tracer diffusive operator

The lateral fourth order tracer diffusive operator is defined by :

$$D^{lT} = \Delta\left(A^{lT}\;\Delta T\right) \qquad \text{where } D^{lT} = \Delta\left(A^{lT}\;\Delta T\right) \tag{2.49}$$

It is the second order operator given by (2.44) applied twice with the eddy diffusion coefficient correctly placed.

lateral second order momentum diffusive operator

The second order momentum diffusive operator along z- or s-surfaces is found by applying (2.19e) to the horizontal velocity vector (see Appendix B) :

$$\begin{aligned} \mathbf{D}^{l\mathbf{U}} &= \quad \nabla_h\left(A^{lm}\chi\right) \;-\; \nabla_h \times \left(A^{lm}\,\zeta\;\mathbf{k}\right) \\ &= \begin{pmatrix} \dfrac{1}{e_1}\dfrac{\partial\left(A^{lm}\chi\right)}{\partial i} - \dfrac{1}{e_2 e_3}\dfrac{\partial\left(A^{lm}\;e_3\zeta\right)}{\partial j} \\ \dfrac{1}{e_2}\dfrac{\partial\left(A^{lm}\chi\right)}{\partial j} + \dfrac{1}{e_1 e_3}\dfrac{\partial\left(A^{lm}\;e_3\zeta\right)}{\partial i} \end{pmatrix} \end{aligned} \tag{2.50}$$

Such a formulation ensures a complete separation between the vorticity and horizontal divergence fields (see Appendix C). Unfortunately, it is not available for geopotential diffusion in $s-$coordinates and for isoneutral diffusion in both z- and s-coordinates ($i.e.$ when a rotation is required). In these two cases, the u and $v-$fields are considered as independent scalar fields, so that the diffusive operator is given by :

$$\begin{aligned} D_u^{l\mathbf{U}} &= \nabla.\left(\Re\;\nabla u\right) \\ D_v^{l\mathbf{U}} &= \nabla.\left(\Re\;\nabla v\right) \end{aligned} \tag{2.51}$$

where $\Re$ is given by (2.44). It is the same expression as those used for diffusive operator on tracers. It must be emphasised that such a formulation is only exact in a Cartesian coordinate system, $i.e.$ on a $f-$ or $\beta-$plane, not on the sphere. It is also a very good approximation in vicinity of the Equator in a geographical coordinate system [**?**].

lateral fourth order momentum diffusive operator

As for tracers, the fourth order momentum diffusive operator along z or s-surfaces is a re-entering second order operator (2.50) or (2.50) with the eddy viscosity coefficient correctly placed :

geopotential diffusion in z-coordinate :

$$\begin{aligned} \mathbf{D}^{l\mathbf{U}} &= \nabla_h \left\{ \nabla_h \cdot \left[A^{lm} \nabla_h \left(\chi \right) \right] \right\} \\ &\quad + \nabla_h \times \left\{ \mathbf{k} \cdot \nabla \times \left[A^{lm} \nabla_h \times \left(\zeta \, \mathbf{k} \right) \right] \right\} \end{aligned} \tag{2.52}$$

geopotential diffusion in s-coordinate :

$$\left\{ \begin{aligned} D_u^{l\mathbf{U}} &= \Delta \left(A^{lm} \, \Delta u \right) \\ D_v^{l\mathbf{U}} &= \Delta \left(A^{lm} \, \Delta v \right) \end{aligned} \right. \quad \text{where} \quad \Delta \left(\bullet \right) = \nabla \cdot \left(\Re \nabla (\bullet) \right) \tag{2.53}$$

Space and Time Domain (DOM)

Contents

Having defined the continuous equations in Chap. 2, we need to choose a discretization on a grid, and numerical algorithms. In the present chapter, we provide a general description of the staggered grid used in *NEMO* , and other information relevant to the main directory routines (time stepping, main program) as well as the DOM (DOMain) directory.

3.1 Fundamentals of the Discretisation

3.1.1 Arrangement of Variables

The numerical techniques used to solve the Primitive Equations in this model are based on the traditional, centred second-order finite difference approximation. Special attention has been given to the homogeneity of the solution in the three space directions. The arrangement of variables is the same in all directions. It consists of cells centred on scalar points (T, S, p, ρ) with vector points (u, v, w) defined in the centre of each face of the cells (Fig. 3.1.1). This is the generalisation to three dimensions of the well-known "C" grid in Arakawa's classification [**?**]. The relative and planetary vorticity, ζ and f, are defined in the centre of each vertical edge and the barotropic stream function ψ is defined at horizontal points overlying the ζ and f-points.

The ocean mesh ($i.e.$ the position of all the scalar and vector points) is defined by the transformation that gives (λ ,φ ,z) as a function of (i, j, k). The grid-points are located at integer or integer and a half value of (i, j, k) as indicated on Table 3.1.1. In all the following, subscripts u, v, w, f, uw, vw or fw indicate the position of the grid-point where the scale factors are defined. Each scale factor is defined as the local analytical value provided by (2.18). As a result, the mesh on which partial derivatives $\frac{\partial}{\partial\lambda}$, $\frac{\partial}{\partial\varphi}$, and $\frac{\partial}{\partial z}$ are evaluated is a uniform mesh with a grid size of unity. Discrete partial derivatives are formulated by the traditional, centred second order finite difference approximation while the scale factors are chosen equal to their local analytical value. An important point here is that the partial derivative of the scale factors must be evaluated by centred finite difference approximation, not from their analytical expression. This preserves the symmetry of the discrete set of equations and therefore satisfies many of the continuous properties (see Appendix C). A similar, related remark can be made about the domain size : when needed, an area, volume, or the total ocean depth must be evaluated as the sum of the relevant scale factors (see (3.6)) in the next section).

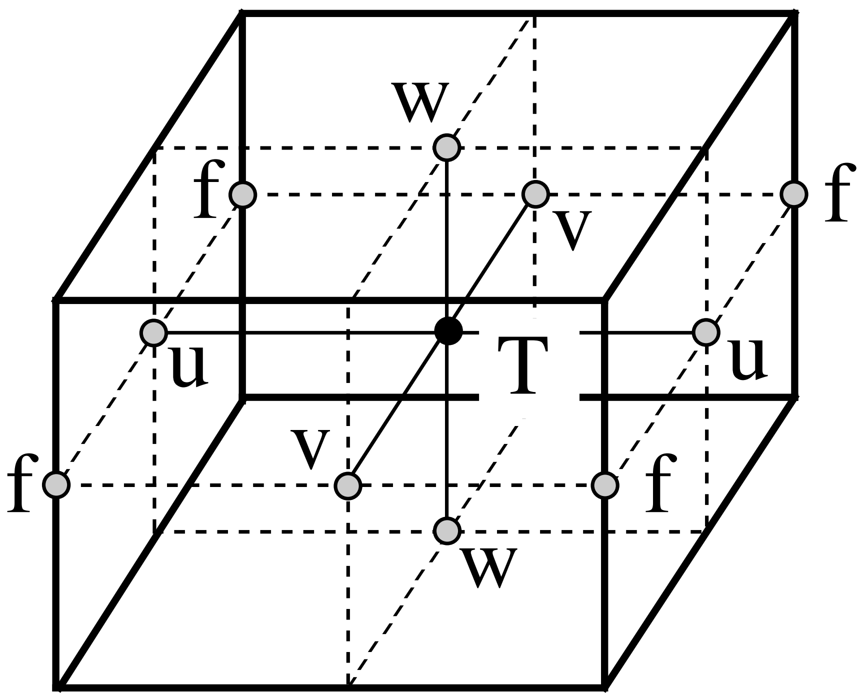


FIG. 3.1 – Arrangement of variables. T indicates scalar points where temperature, salinity, density, pressure and horizontal divergence are defined. (u,v,w) indicates vector points, and f indicates vorticity points where both relative and planetary vorticities are defined

3.1.2 Discrete Operators

Given the values of a variable q at adjacent points, the differencing and averaging operators at the midpoint between them are :

$$\delta_i[q] = \quad q(i+1/2) - q(i-1/2) \qquad (3.1a)$$

$$\overline{q}^{\,i} = \{q(i+1/2) + q(i-1/2)\} \;\; / \; 2 \qquad (3.1b)$$

Similar operators are defined with respect to $i + 1/2$, j, $j + 1/2$, k, and $k + 1/2$. Following (2.19a) and (2.19d), the gradient of a variable q defined at a T-point has its three components defined at u-, v- and w-points while its Laplacien is defined at T-point.

T	i	j	k
u	$i+1/2$	j	k
v	i	$j+1/2$	k
w	i	j	$k+1/2$
f	$i+1/2$	$j+1/2$	k
uw	$i+1/2$	j	$k+1/2$
vw	i	$j+1/2$	$k+1/2$
fw	$i+1/2$	$j+1/2$	$k+1/2$

TAB. 3.1 – Location of grid-points as a function of integer or integer and a half value of the column, line or level. This indexing is only used for the writing of the semi- discrete equation. In the code, the indexing uses integer values only and has a reverse direction in the vertical (see §3.1.3)

These operators have the following discrete forms in the curvilinear s-coordinate system :

$$\nabla q \equiv \frac{1}{e_{1u}} \delta_{i+1/2}\left[q\right] \;\; \mathbf{i} + \frac{1}{e_{2v}} \delta_{j+1/2}\left[q\right] \;\; \mathbf{j} + \frac{1}{e_{3w}} \delta_{k+1/2}\left[q\right] \;\; \mathbf{k} \tag{3.2}$$

$$\begin{aligned}\Delta q \equiv \frac{1}{e_{1T}\,e_{2T}\,e_{3T}} \;\; & \left(\delta_i \left[\frac{e_{2u}e_{3u}}{e_{1u}} \; \delta_{i+1/2}\left[q\right] \right] + \delta_j \left[\frac{e_{1v}e_{3v}}{e_{2v}} \; \delta_{j+1/2}\left[q\right] \right] \right) \\ & + \frac{1}{e_{3T}} \delta_k \left[\frac{1}{e_{3w}} \; \delta_{k+1/2}\left[q\right] \right]\end{aligned} \tag{3.3}$$

Following (2.19c) and (2.19b), a vector $\mathbf{A} = (a_1, a_2, a_3)$ defined at vector points (u, v, w) has its three curl components defined at vw-, uw, and f-points, and its divergence defined at T-points :

$$\begin{aligned}\nabla \times \mathbf{A} \equiv & \frac{1}{e_{2v}\,e_{3vw}} \;\; \big(\delta_{j+1/2}\left[e_{3w}a_3\right] - \delta_{k+1/2}\left[e_{2v}a_2\right] \big) \;\; \mathbf{i} \\ & + \frac{1}{e_{2u}\,e_{3uw}} \;\; \big(\delta_{k+1/2}\left[e_{1u}a_1\right] - \delta_{i+1/2}\left[e_{3w}a_3\right] \big) \;\; \mathbf{j} \\ & + \frac{e_{3f}}{e_{1f}\,e_{2f}} \;\; \big(\delta_{i+1/2}\left[e_{2v}a_2\right] - \delta_{j+12}\left[e_{1u}a_1\right] \big) \;\; \mathbf{k}\end{aligned} \tag{3.4}$$

$$\nabla \cdot \mathbf{A} = \frac{1}{e_{1T}e_{2T}e_{3T}} \left(\delta_i \left[e_{2u}e_{3u}a_1\right] + \delta_j \left[e_{1v}e_{3v}a_2\right] \right) + \frac{1}{e_{3T}} \delta_k \left[a_3\right] \tag{3.5}$$

In the special case of a pure z-coordinate system, (3.3) and (3.5) can be simplified. In this case, the vertical scale factor becomes a function of the single variable k and thus does not depend on the horizontal location of a grid point. For example (3.5) reduces to :

$$\nabla \cdot \mathbf{A} = \frac{1}{e_{1T}e_{2T}} \left(\delta_i \left[e_{2u}a_1\right] + \delta_j \left[e_{1v}a_2\right] \right) + \frac{1}{e_{3T}} \delta_k \left[a_3\right]$$

The vertical average over the whole water column denoted by an overbar becomes for a quantity q which is a masked field (i.e. equal to zero inside solid area) :

$$\bar{q} = \frac{1}{H} \int_{k^b}^{k^o} q \; e_{3q} \, dk \equiv \frac{1}{H_q} \sum_k q \; e_{3q} \tag{3.6}$$

where H_q is the ocean depth, which is the masked sum of the vertical scale factors at q points, k^b and k^o are the bottom and surface k-indices, and the symbol k^o refers to a summation over all grid points of the same type in the direction indicated by the subscript (here k).

In continuous form, the following properties are satisfied :

$$\nabla \times \nabla q = \mathbf{0} \tag{3.7}$$

$$\nabla \cdot (\nabla \times \mathbf{A}) = 0 \tag{3.8}$$

It is straightforward to demonstrate that these properties are verified locally in discrete form as soon as the scalar q is taken at T-points and the vector **A** has its components defined at vector points (u, v, w).

Let a and b be two fields defined on the mesh, with value zero inside continental area. Using integration by parts it can be shown that the differencing operators (δ_i, δ_j and δ_k) are anti-symmetric linear operators, and further that the averaging operators $\overline{\cdot}^{\,i}$, $\overline{\cdot}^{\,k}$ and $\overline{\cdot}^{\,k}$) are symmetric linear operators, $i.e.$

$$\sum_i a_i \; \delta_i \left[b \right] \equiv - \sum_i \delta_{i+1/2} \left[a \right] \; b_{i+1/2} \tag{3.9}$$

$$\sum_i a_i \; \overline{b}^{\,i} \equiv \sum_i \overline{a}^{\,i+1/2} \; b_{i+1/2} \tag{3.10}$$

In other words, the adjoint of the differencing and averaging operators are $\delta_i^* = \delta_{i+1/2}$ and $(\overline{\cdot}^{\,i})^* = \overline{\cdot}^{\,i+1/2}$, respectively. These two properties will be used extensively in the Appendix C to demonstrate integral conservative properties of the discrete formulation chosen.

3.1.3 Numerical Indexing

The array representation used in the FORTRAN code requires an integer indexing while the analytical definition of the mesh (see §3.1.1) is associated with the use of integer values for T-points and both integer and integer and a half values for all the other points. Therefore a specific integer indexing must be defined for points other than T-points ($i.e.$ velocity and vorticity grid-points). Furthermore, the direction of the vertical indexing has been changed so that the surface level is at $k = 1$.

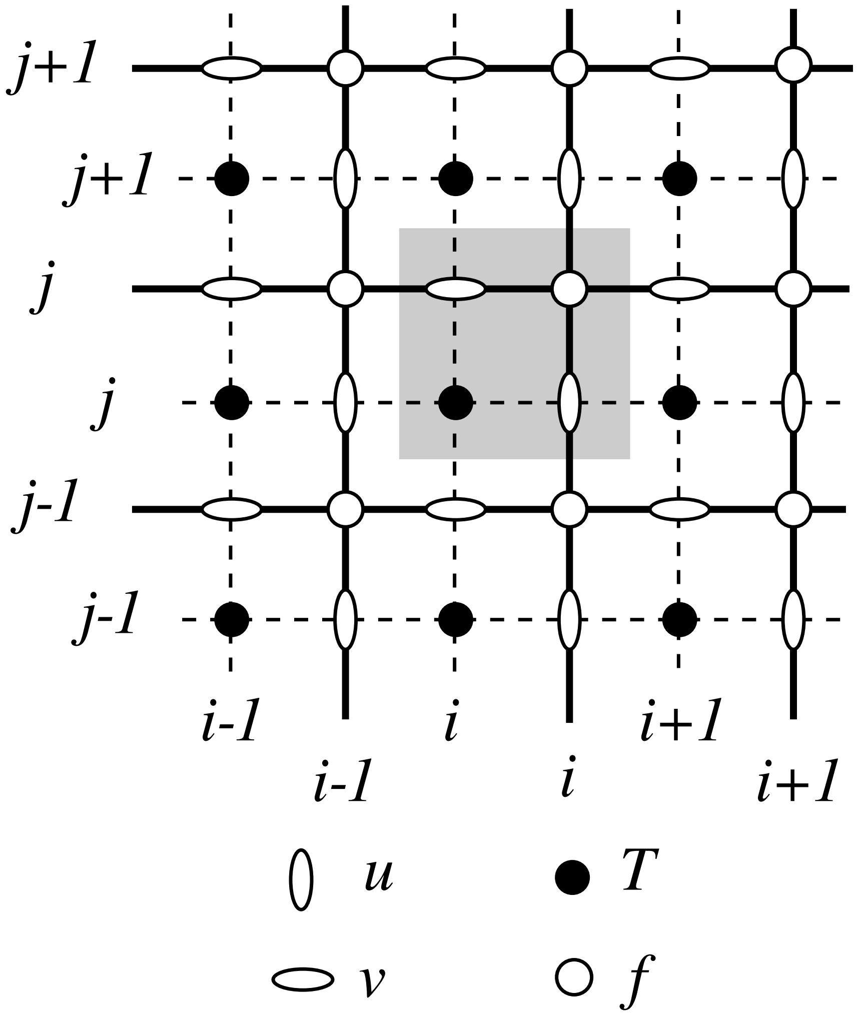


FIG. 3.2 – Horizontal integer indexing used in the FORTRAN code. The dashed area indicates the cell in which variables contained in arrays have the same i- and j-indices

Horizontal Indexing

The indexing in the horizontal plane has been chosen as shown in Fig.3.1.3. For an increasing i index (j index), the T-point and the eastward u-point (northward v-point) have the same index (see the dashed area in Fig.3.1.3). A T-point and its nearest northeast f-point have the same i-and j-indices.

Vertical Indexing

In the vertical, the chosen indexing requires special attention since the k-axis is re-orientated downward in the FORTRAN code compared to the indexing used in the semi-discrete equations and given in §3.1.1. The sea surface corresponds to the w-level $k = 1$ which is the same index as T-level just below (Fig.3.1.3). The last w-level ($k = jpk$) either corresponds to the ocean floor or is inside the bathymetry while the last T-level is always inside the bathymetry (Fig.3.1.3). Note that for an increasing k index, a w-point and the T-point just below have the same k index, in opposition to what is done in the horizontal plane where it is the T-point and the nearest velocity points in the direction of the horizontal axis that have the same i or j index (compare the dashed area in Fig.3.1.3 and 3.1.3). Since the scale factors are chosen to be strictly positive, a *minus sign* appears in the FORTRAN code *before all the vertical derivatives* of the discrete equations given in this documentation.

Domain Size

The total size of the computational domain is set by the parameters *jpiglo*, *jpjglo* and *jpk* in the i, j and k directions respectively. They are given as parameters in the *par_oce.F90* module[1]. The use of parameters rather than variables (together with dynamic allocation of arrays) was chosen because it ensured that the compiler would optimize the executable code efficiently, especially on vector machines (optimization may be less efficient when the problem size is unknown at the time of compilation). Nevertheless, it is possible to set up the code with full dynamical allocation by using the AGRIF packaged [**?**]. Note that are other parameters in *par_oce.F90* that refer to the domain size. The two parameters $jpidta$ and $jpjdta$ may be larger than $jpiglo$, $jpjglo$ when the user wants to use only a sub-region of a given configuration. This is the "zoom" capability described in §10.3. In most applications of the model, $jpidta = jpiglo$, $jpjdta = jpjglo$, and $jpizoom = jpjzoom = 1$. Parameters jpi and jpj refer to the size of each processor subdomain when the code is run in parallel using domain decomposition (**key_mpp_mpi** defined, see §7.3).

3.2 Domain : Horizontal Grid (mesh) (*domhgr.F90* module)

3.2.1 Coordinates and scale factors

The ocean mesh ($i.e.$ the position of all the scalar and vector points) is defined by the transformation that gives (λ, φ, z) as a function of (i, j, k). The grid-points are located at integer or integer and a half values of as indicated in Table 3.1.1. The associated scale factors are defined using the analytical first derivative of the transformation (2.18). These

[1]When a specific configuration is used (ORCA2 global ocean, etc...) the parameter are actually defined in additional files introduced by *par_oce.F90* module via CPP *include* command. For example, ORCA2 parameters are set in *par_ORCA_R2.h90* file

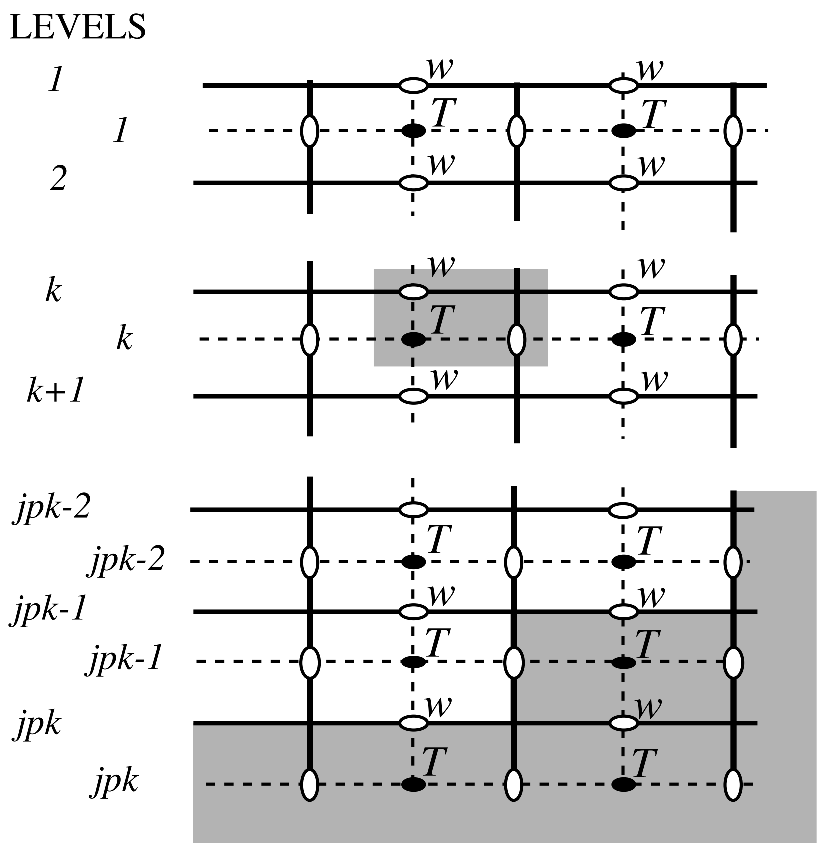


FIG. 3.3 – Vertical integer indexing used in the FORTRAN code. Note that the k-axis is orientated downward. The dashed area indicates the cell in which variables contained in arrays have the same k-index.

definitions are done in two modules, *domhgr.F90* and *domzgr.F90*, which provide the horizontal and vertical meshes, respectively. This section deals with the horizontal mesh parameters.

In a horizontal plane, the location of all the model grid points is defined from the analytical expressions of the longitude λ and latitude φ as a function of (i, j). The horizontal scale factors are calculated using (2.18). For example, when the longitude and latitude are function of a single value (i and j, respectively) (geographical configuration of the mesh), the horizontal mesh definition reduces to define the wanted $\lambda(i)$, $\varphi(j)$, and their derivatives $\lambda'(i)$ $\varphi'(j)$ in the *domhgr.F90* module. The model computes the grid-point positions and scale factors in the horizontal plane as follows :

$$\begin{aligned}
\lambda_T &\equiv \text{glamt} = \lambda(i) & \varphi_T &\equiv \text{gphit} = \varphi(j) \\
\lambda_u &\equiv \text{glamu} = \lambda(i+1/2) & \varphi_u &\equiv \text{gphiu} = \varphi(j) \\
\lambda_v &\equiv \text{glamv} = \lambda(i) & \varphi_v &\equiv \text{gphiv} = \varphi(j+1/2) \\
\lambda_f &\equiv \text{glamf } = \lambda(i+1/2) & \varphi_f &\equiv \text{gphif } = \varphi(j+1/2)
\end{aligned}$$

$$\begin{aligned}
e_{1T} &\equiv \text{e1t} = r_a |\lambda'(i) \ \cos\varphi(j)| & e_{2T} &\equiv \text{e2t} = r_a |\varphi'(j)| \\
e_{1u} &\equiv \text{e1t} = r_a |\lambda'(i+1/2) \ \cos\varphi(j)| & e_{2u} &\equiv \text{e2t} = r_a |\varphi'(j)| \\
e_{1v} &\equiv \text{e1t} = r_a |\lambda'(i) \ \cos\varphi(j+1/2)| & e_{2v} &\equiv \text{e2t} = r_a |\varphi'(j+1/2)| \\
e_{1f} &\equiv \text{e1t} = r_a |\lambda'(i+1/2) \ \cos\varphi(j+1/2)| & e_{2f} &\equiv \text{e2t} = r_a |\varphi'(j+1/2)|
\end{aligned}$$

where the last letter of each computational name indicates the grid point considered and r_a is the earth radius (defined in *phycst.F90* along with all universal constants). Note that the horizontal position of and scale factors at w-points are exactly equal to those of T-points, thus no specific arrays are defined at w-points.

Note that the definition of the scale factors ($i.e.$ as the analytical first derivative of the transformation that gives (λ, φ, z) as a function of (i, j, k)) is specific to the *NEMO* model [**?**]. As an example, e_{1T} is defined locally at a T-point, whereas many other models on a C grid choose to define such a scale factor as the distance between the U-points on each side of the T-point. Relying on an analytical transformation has two advantages : firstly, there is no ambiguity in the scale factors appearing in the discrete equations, since they are first introduced in the continuous equations ; secondly, analytical transformations encourage good practice by the definition of smoothly varying grids (rather than allowing the user to set arbitrary jumps in thickness between adjacent layers) [**?**]. An example of the effect of such a choice is shown in Fig. 3.2.1.

3.2.2 Choice of horizontal grid

The user has three options available in defining a horizontal grid, which involve the parameter $jphgr\_mesh$ of the *par_oce.F90* module.

***jphgr_mesh*=0** The most general curvilinear orthogonal grids. The coordinates and their first derivatives with respect to i and j are provided in a file, read in *hgr_read* subroutine of the domhgr module.

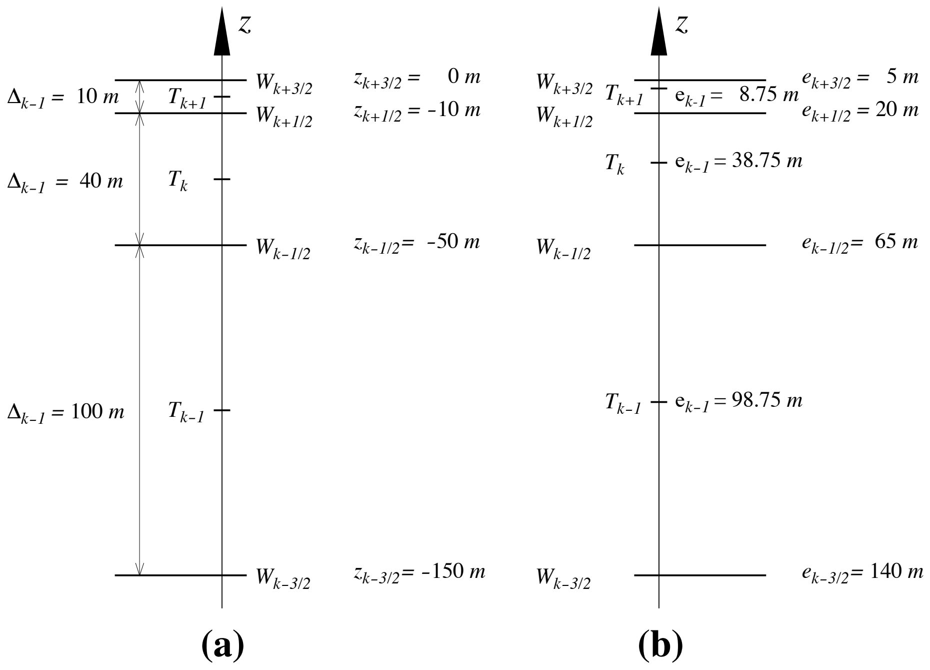


FIG. 3.4 – Comparison of (a) traditional definitions of grid-point position and grid-size in the vertical, and (b) analytically derived grid-point position and scale factors. For both grids here, the same w-point depth has been chosen but in (a) the T-points are set half way between w-points while in (b) they are defined from an analytical function : $z(k) = 5\,(i-1/2)^3 - 45\,(i-1/2)^2 + 140\,(i-1/2) - 150$. Note the resulting difference between the value of the grid-size Δ_k and those of the scale factor e_k.

***jphgr_mesh*=1 to 5** A few simple analytical grids are provided (see below). For other analytical grids, the *domhgr.F90* module must be modified by the user.

There are two simple cases of geographical grids on the sphere. With *jphgr_mesh*=1, the grid (expressed in degrees) is regular in space, with grid sizes specified by parameters *ppe1_deg* and *ppe2_deg*, respectively. Such a geographical grid can be very anisotropic at high latitudes because of the convergence of meridians (the zonal scale factors e_1 become much smaller than the meridional scale factors e_2). The Mercator grid (*jphgr_mesh*=4) avoids this anisotropy by refining the meridional scale factors in the same way as the zonal ones. In this case, meridional scale factors and latitudes are calculated analytically using the formulae appropriate for a Mercator projection, based on *ppe1_deg* which is a reference grid spacing at the equator (this applies even when the geographical equator is situated outside the model domain). In these two cases (*jphgr_mesh*=1 or 4), the grid position is defined by the longitude and latitude of the south-westernmost point (*ppglamt0* and *ppgphi0*). Note that for the Mercator grid the user need only provide an approximate starting latitude : the real latitude will be recalculated analytically, in order to ensure that the equator corresponds to line passing through T- and u-points.

Rectangular grids ignoring the spherical geometry are defined with *jphgr_mesh* = 2, 3, 5. The domain is either an f-plane (*jphgr_mesh* = 2, Coriolis factor is constant) or a beta-plane (*jphgr_mesh* = 3, the Coriolis factor is linear in the j-direction). The grid size is uniform in meter in each direction, and given by the parameters *ppe1_m* and *ppe2_m* respectively. The zonal grid coordinate (*glam* arrays) is in kilometers, starting at zero with the first T-point. The meridional coordinate (gphi. arrays) is in kilometers, and the second T-point corresponds to coordinate $gphit = 0$. The input parameter *ppglam0* is ignored. *ppgphi0* is used to set the reference latitude for computation of the Coriolis parameter. In the case of the beta plane, *ppgphi0* corresponds to the center of the domain. Finally, the special case *jphgr_mesh*=5 corresponds to a beta plane in a rotated domain for the GYRE configuration, representing a classical mid-latitude double gyre system. The rotation allows us to maximize the jet length relative to the gyre areas (and the number of grid points).

The choice of the grid must be consistent with the boundary conditions specified by the parameter *jperio* (see §7).

3.2.3 Grid files

All the arrays relating to a particular ocean model configuration (grid-point position, scale factors, masks) can be saved in files if *nmsh* $\neq 0$ (namelist parameter). This can be particularly useful for plots and off-line diagnostics. In some cases, the user may choose to make a local modification of a scale factor in the code. This is the case in global configurations when restricting the width of a specific strait (usually a one-grid-point strait that happens to be too wide due to insufficient model resolution). An example is Gibraltar Strait in the ORCA2 configuration. When such modifications are done, the output grid written when *nmsh* $\neq 0$ is no more equal to the input grid.

3.3 Domain : Vertical Grid (*domzgr.F90* module)

```
!-----------------------------------------------------------------------
&nam_zgr       !   vertical coordinate
!-----------------------------------------------------------------------
   ln_zco      = .false.   !  z-coordinate - full    steps   (T/F)      ("key_zco" may also be defined)
   ln_zps      = .true.    !  z-coordinate - partial steps   (T/F)
   ln_sco      = .false.   !  s- or hybrid z-s-coordinate    (T/F)
/
```

```
!-----------------------------------------------------------------------
&namdom        !   space and time domain (bathymetry, mesh, timestep)
!-----------------------------------------------------------------------
   ntopo       =    1      !  compute (=0) or read(=1) the bathymetry file
   e3zps_min   =    5.     !  the thickness of the partial step is set larger than the minimum
   e3zps_rat   =    0.1    !  of e3zps_min and e3zps_rat * e3t   (N.B. 0<e3zps_rat<1)
   nmsh        =    0      !  create (=1) a mesh file (coordinates, scale factors, masks) or not (=0)
   nacc        =    0      !  =1 acceleration of convergence method used, rdt < rdttra(k)
                           !  =0, no acceleration, rdt = rdttra
   atfp        =    0.1    !  asselin time filter parameter
   rdt         = 5760.     !  time step for the dynamics (and tracer if nacc=0)
   rdtmin      = 5760.     !  minimum time step on tracers (used if nacc=1)
   rdtmax      = 5760.     !  maximum time step on tracers (used if nacc=1)
```

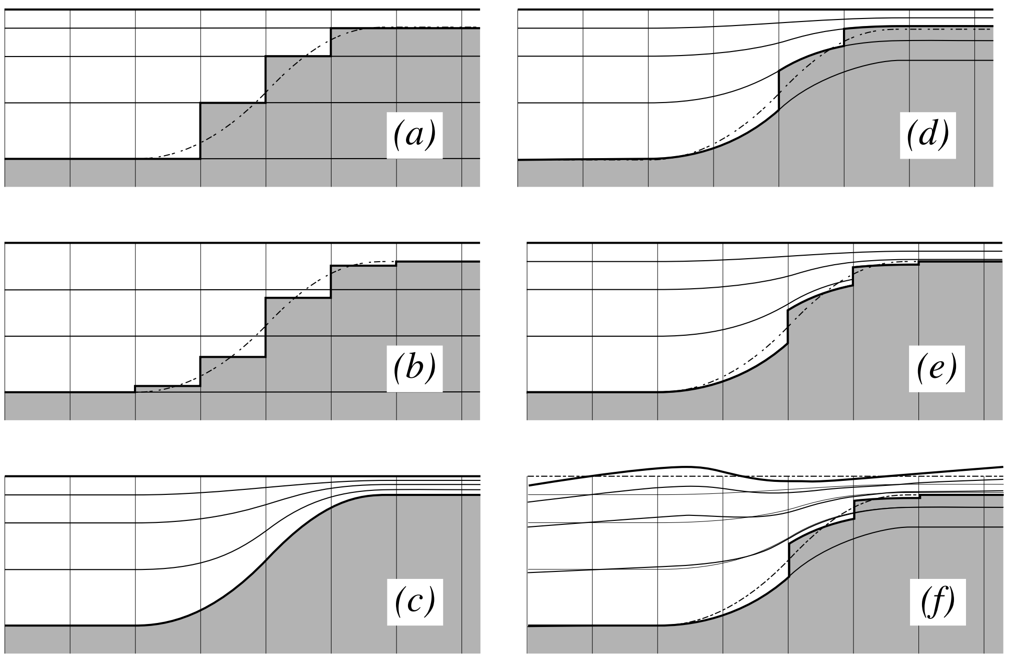


FIG. 3.5 – The ocean bottom as seen by the model : (a) z-coordinate with full step, (b) z-coordinate with partial step, (c) s-coordinate : terrain following representation, (d) hybrid $s - z$ coordinate, (e) hybrid $s - z$ coordinate with partial step, and (f) same as (e) but with variable volume associated with the non-linear free surface. Note that the variable volume option (**key_vvl**) can be used with any of the 5 coordinates (a) to (e).

```
   rdth        =  800.     !  depth variation of tracer time step  (used if nacc=1)
   rdtbt       =   90.     !  barotropic time step (for the split explicit algorithm) ("key_dynspg_ts")
   nclosea     =    0      !  = 0 no closed sea in the model domain
                           !  = 1 closed sea (Black Sea, Caspian Sea, Great US Lakes...)
/
```

In the vertical, the model mesh is determined by four things : (1) the bathymetry given in meters ; (2) the number of levels of the model (*jpk*) ; (3) the analytical transformation $z(i, j, k)$ and the vertical scale factors (derivatives of the transformation) ; and (4) the masking system, $i.e.$ the number of wet model levels at each (i, j) column of points.

The choice of a vertical coordinate, even if it is made through a namelist parameter, must be done once of all at the beginning of an experiment. It is not intended as an option which can be enabled or disabled in the middle of an experiment. Three main choices are offered (Fig. 3.3a to c) : z-coordinate with full step bathymetry (*ln_zco*=true), z-coordinate with partial step bathymetry (*ln_zps*=true), or generalized, s-coordinate (*ln_sco*=true). Hybridation of the three main coordinates are available : $s - z$ or $s - zps$ coordinate (Fig. 3.3d and 3.3e). When using the variable volume option **key_vvl**) ($i.e.$ non-linear free surface), the coordinate follow the time-variation of the free surface so that the transformation is time dependent : $z(i,j,k,t)$ (Fig. 3.3f). This option can be used with full step bathymetry or s-coordinate (hybride and partial step coordinates have not yet been tested in NEMO v2.3).

Contrary to the horizontal grid, the vertical grid is computed in the code and no provision is made for reading it from a file. The only input file is the bathymetry (in meters)[2]. After reading the bathymetry, the algorithm for vertical grid definition differs between the different options :

zco set a reference coordinate transformation $z_0(k)$, and set $z(i,j,k,t) = z_0(k)$.

zps set a reference coordinate transformation $z_0(k)$, and calculate the thickness of the deepest level at each (i,j) point using the bathymetry, to obtain the final three-dimensional depth and scale factor arrays.

sco smooth the bathymetry to fulfil the hydrostatic consistency criteria and set the three-dimensional transformation.

s-z* and *s-zps smooth the bathymetry to fulfil the hydrostatic consistency criteria and set the three-dimensional transformation $z(i,j,k)$, and possibly introduce masking of extra land points to better fit the original bathymetry file

Generally, the arrays describing the grid point depths and vertical scale factors are three dimensional arrays (i,j,k). In the special case of z-coordinates with full step bottom topography, it is possible to define those arrays as one-dimensional, in order to save memory. This is performed by defining the **key_zco** C-Pre-Processor (CPP) key. To improve the code readability while providing this flexibility, the vertical coordinate and scale factors are defined as functions of (i,j,k) with "fs" as prefix (examples : *fsdeptht*, *fse3t*, etc) that can be either three-dimensional arrays, or a one dimensional array when **key_zco** is defined. These functions are defined in the file *domzgr_substitute.h90* of the DOM directory. They are used throughout the code, and replaced by the corresponding arrays at the time of pre-processing (CPP capability).

3.3.1 Meter Bathymetry

Three options are possible for defining the bathymetry, according to the namelist variable *ntopo* :

***ntopo* = 0** a flat-bottom domain is defined. The total depth $z_w(jpk)$ is given by the coordinate transformation. The domain can either be a closed basin or a periodic channel depending on the parameter *jperio*.

***ntopo* = -1** a domain with a bump of topography one third of the domain width at the central latitude. This is meant for the "EEL-R5" configuration, a periodic or open boundary channel with a seamount.

***ntopo* = 1** read a bathymetry. The bathymetry file (Netcdf format) provides the ocean depth (positive, in meters) at each grid point of the model grid. The bathymetry is

[2]N.B. in full step z-coordinate, a *bathy_level* file can replace the *bathy_meter* file, so that the computation of the number of wet ocean point in each water column is by-passed

usually built by interpolating a standard bathymetry product ($e.g.$ ETOPO2) onto the horizontal ocean mesh. Defining the bathymetry also defines the coastline : where the bathymetry is zero, no model levels are defined (all levels are masked).

When using the rigid lid approximation (**key_dynspg_rl** is defined) isolated land masses (islands) must be identified by negative integers in the input bathymetry file (see §10.7.4).

When a global ocean is coupled to an atmospheric model it is better to represent all large water bodies (e.g, great lakes, Caspian sea...) even if the model resolution does not allow their communication with the rest of the ocean. This is unnecessary when the ocean is forced by fixed atmospheric conditions, so these seas can be removed from the ocean domain. The user has the option to set the bathymetry in closed seas to zero (see §10.2), but the code has to be adapted to the user's configuration.

3.3.2 z-coordinate (*ln_zco*=.true. or key_zco) and reference coordinate

The reference coordinate transformation $z_0(k)$ defines the arrays $gdept_0$ and $gdepw_0$ for T- and w-points, respectively. As indicated on Fig.3.1.3 *jpk* is the number of w-levels. $gdepw_0(1)$ is the ocean surface. There are at most *jpk*-1 T-points inside the ocean, the additional T-point at $jk = jpk$ is below the sea floor and is not used. The vertical location of w- and T-levels is defined from the analytic expression of the depth $z_0(k)$ whose analytical derivative with respect to k provides the vertical scale factors. The user must provide the analytical expression of both z_0 and its first derivative with respect to k. This is done in routine *domzgr.F90* through statement functions, using parameters provided in the *par_oce.h90* file.

It is possible to define a simple regular vertical grid by giving zero stretching (*ppacr=0*). In that case, the parameters *jpk* (number of w-levels) and *pphmax* (total ocean depth in meters) fully define the grid.

For climate-related studies it is often desirable to concentrate the vertical resolution near the ocean surface. The following function is proposed as a standard for a z-coordinate (with either full or partial steps) :

$$
\begin{split}
z_0(k) &= h_{sur} - h_0\, k - \; h_1 \; \log \left[\, \cosh \left((k - h_{th}) / h_{cr} \right) \,\right] \\
e_3^0(k) &= \left| -h_0 - h_1 \; \tanh \left((k - h_{th}) / h_{cr} \right) \right|
\end{split}
\tag{3.11}
$$

where $k = 1$ to *jpk* for w-levels and $k = 1$ to $k = 1$ for $T-$levels. Such an expression allows us to define a nearly uniform vertical location of levels at the ocean top and bottom with a smooth hyperbolic tangent transition in between (Fig. 3.3.2).

The most used vertical grid for ORCA2 has $10\ m$ ($500\ m$) resolution in the surface (bottom) layers and a depth which varies from 0 at the sea surface to a minimum of

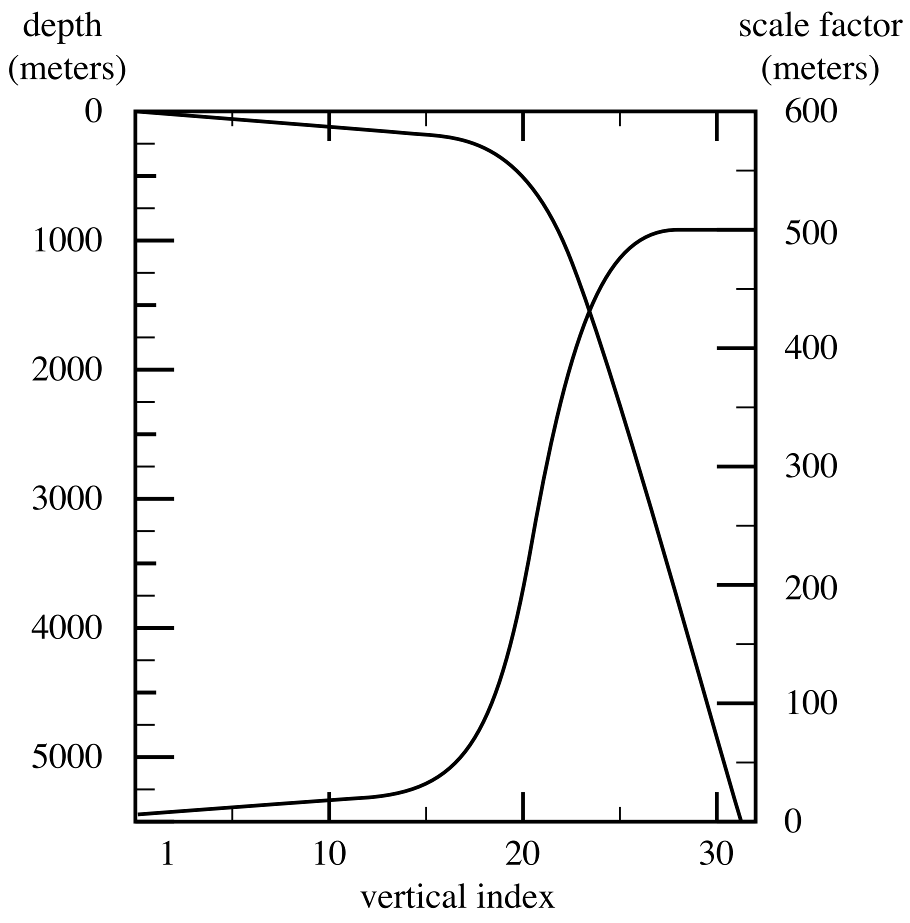


FIG. 3.6 – Default vertical mesh for ORCA2 : 30 ocean levels (L30). Vertical level functions for (a) T-point depth and (b) the associated scale factor as computed from (3.11) using (3.12) in z-coordinate.

$-5000\ m$. This leads to the following conditions :

$$\begin{aligned} e_3(1+1/2) &= 10. \\ e_3(jpk-1/2) &= 500. \\ z(1) &= 0. \\ z(jpk) &= -5000. \end{aligned} \tag{3.12}$$

With the choice of the stretching $h_{cr} = 3$ and the number of levels *jpk*=31, the four coefficients h_{sur}, h_0, h_1, and h_{th} in (3.11) have been determined such that (3.12) is satisfied, through an optimisation procedure using a bisection method. For the first standard

ORCA2 vertical grid this led to the following values : $h_{sur} = 4762.96$, $h_0 = 255.58$, $h_1 = 245.5813$, and $h_{th} = 21.43336$. The resulting depths and scale factors as a function of the model levels are shown in Fig. 3.3.2 and given in Table 3.3.2. Those values correspond to the parameters *ppsur*, *ppa0*, *ppa1*, *ppkth* in the parameter file *par_oce.F90*.

Rather than entering parameters h_{sur}, h_0, and h_1 directly, it is possible to recalculate them. In that case the user sets *ppsur=ppa0=ppa1=pp_to_be_computed*, in *par_oce.F90*, and specifies instead the four following parameters :

- *ppacr*=h_{cr} : stretching factor (nondimensional). The larger *ppacr*, the smaller the stretching. Values from 3 to 10 are usual.
- *ppkth*=h_{th} : is approximately the model level at which maximum stretching occurs (nondimensional, usually of order 1/2 or 2/3 of *jpk*)
- *ppdzmin* : minimum thickness for the top layer (in meters)
- *pphmax* : total depth of the ocean (meters).

As an example, for the 45 layers used in the DRAKKAR configuration those parameters are : *jpk*=46, *ppacr*=9, *ppkth*=23.563, *ppdzmin*=6m, *pphmax*=5750m.

3.3.3 z-coordinate with partial step (*ln_zps*=.true.)

```
!-----------------------------------------------------------------------
&namdom       !   space and time domain (bathymetry, mesh, timestep)
!-----------------------------------------------------------------------
   ntopo      =    1      !  compute (=0) or read(=1) the bathymetry file
   e3zps_min  =    5.     !  the thickness of the partial step is set larger than the minimum
   e3zps_rat  =    0.1    !  of e3zps_min and e3zps_rat * e3t   (N.B. 0<e3zps_rat<1)
   nmsh       =    0      !  create (=1) a mesh file (coordinates, scale factors, masks) or not (=0)
   nacc       =    0      !  =1 acceleration of convergence method used, rdt < rdttra(k)
                          !  =0, no acceleration, rdt = rdttra
   atfp       =    0.1    !  asselin time filter parameter
   rdt        = 5760.     !  time step for the dynamics (and tracer if nacc=0)
   rdtmin     = 5760.     !  minimum time step on tracers (used if nacc=1)
   rdtmax     = 5760.     !  maximum time step on tracers (used if nacc=1)
   rdth       =  800.     !  depth variation of tracer time step  (used if nacc=1)
   rdtbt      =   90.     !  barotropic time step (for the split explicit algorithm) ("key_dynspg_ts")
   nclosea    =    0      !  = 0 no closed sea in the model domain
                          !  = 1 closed sea (Black Sea, Caspian Sea, Great US Lakes...)
/
```

In z-coordinate partial step, the depths of the model levels are defined by the reference analytical function $z_0(k)$ as described in the previous section, *except* in the bottom layer. The thickness of the bottom layer is allowed to vary as a function of geographical location (λ, φ) to allow a better representation of the bathymetry, especially in the case of small slopes (where the bathymetry varies by less than one level thickness from one grid point to the next). The reference layer thicknesses e_{3t}^0 have been defined in the absence of bathymetry. With partial steps, layers from 1 to *jpk*-2 can have a thickness smaller than $e_{3t}(jk)$. The model deepest layer (*jpk*-1) is allowed to have either a smaller or larger thickness than $e_{3t}(jpk)$: the maximum thickness allowed is $2 * e_{3t}(jpk-1)$. This has to be kept in mind when specifying the maximum depth *pphmax* in partial steps : for example, with *pphmax*$=5750\ m$ for the DRAKKAR 45 layer grid, the maximum ocean depth allowed is actually $6000\ m$ (the default thickness $e_{3t}(jpk-1)$ being $250\ m$). Two variables in the namdom namelist are used to define the partial step vertical grid. The mimimum water thickness (in meters) allowed for a cell partially filled with bathymetry at level jk is the minimum of *e3zpsmin* (thickness in meters, usually $20\ m$) or $e_{3t}(jk) *$ *e3zps_rat* (a fraction, usually 10%, of the default thickness $e_{3t}(jk)$).

LEVEL	GDEPT	GDEPW	E3T	E3W
1	**5.00**	0.00	**10.00**	10.00
2	**15.00**	10.00	**10.00**	10.00
3	**25.00**	20.00	**10.00**	10.00
4	**35.01**	30.00	**10.01**	10.00
5	**45.01**	40.01	**10.01**	10.01
6	**55.03**	50.02	**10.02**	10.02
7	**65.06**	60.04	**10.04**	10.03
8	**75.13**	70.09	**10.09**	10.06
9	**85.25**	80.18	**10.17**	10.12
10	**95.49**	90.35	**10.33**	10.24
11	**105.97**	100.69	**10.65**	10.47
12	**116.90**	111.36	**11.27**	10.91
13	**128.70**	122.65	**12.47**	11.77
14	**142.20**	135.16	**14.78**	13.43
15	**158.96**	150.03	**19.23**	16.65
16	**181.96**	169.42	**27.66**	22.78
17	**216.65**	197.37	**43.26**	34.30
18	**272.48**	241.13	**70.88**	55.21
19	**364.30**	312.74	**116.11**	90.99
20	**511.53**	429.72	**181.55**	146.43
21	**732.20**	611.89	**261.03**	220.35
22	**1033.22**	872.87	**339.39**	301.42
23	**1405.70**	1211.59	**402.26**	373.31
24	**1830.89**	1612.98	**444.87**	426.00
25	**2289.77**	2057.13	**470.55**	459.47
26	**2768.24**	2527.22	**484.95**	478.83
27	**3257.48**	3011.90	**492.70**	489.44
28	**3752.44**	3504.46	**496.78**	495.07
29	**4250.40**	4001.16	**498.90**	498.02
30	**4749.91**	4500.02	**500.00**	499.54
31	**5250.23**	5000.00	**500.56**	500.33

TAB. 3.2 – Default vertical mesh in z-coordinate for 30 layers ORCA2 configuration as computed from (3.11) using the coefficients given in (3.12)

Add a figure here of pstep especially at last ocean level

3.3.4 s-coordinate (*ln_sco*=true)

```
!-----------------------------------------------------------------------
&nam_zgr_sco   !   s-coordinate or hybrid z-s-coordinate
!-----------------------------------------------------------------------
   sbot_min    =  300.     !  minimum depth of s-bottom surface (>0) (m)
   sbot_max    = 5250.     !  maximum depth of s-bottom surface (= ocean depth) (>0) (m)
   theta       =    6.0    !  surface control parameter (0<=theta<=20)
   thetb       =    0.75   !  bottom control parameter  (0<=thetb<= 1)
   r_max       =    0.15   !  maximum cut-off r-value allowed (0<r_max<1)
/
```

In s-coordinate (**key_sco** is defined), the depth and thickness of the model levels are defined from the product of a depth field and either a stretching function or its derivative, respectively :

$$\begin{aligned} z(k) &= h(i,j)\; z_0(k) \\ e_3(k) &= h(i,j)\; z_0'(k) \end{aligned} \tag{3.13}$$

where h is the depth of the last w-level ($z_0(k)$) defined at the T-point location in the horizontal and $z_0(k)$ is a function which varies from 0 at the sea surface to 1 at the ocean bottom. The depth field h is not necessary the ocean depth, since a mixed step-like and bottom-following representation of the topography can be used (Fig. 3.3d-e). In the example provided (*zgr_s.h90* file) h is a smooth envelope bathymetry and steps are used to represent sharp bathymetric gradients.

A new flexible stretching function, modified from **?** is provided as an example :

$$\begin{aligned} z &= h_c + (h - h_c)\; cs) \\ c(s) &= \frac{[\tanh\left(\theta\,(s+b)\right) - \tanh\left(\theta\, b\right)]}{2\;\sinh\left(\theta\right)} \end{aligned} \tag{3.14}$$

where h_c is the thermocline depth and θ and b are the surface and bottom control parameters such that $0 \leqslant \theta \leqslant 20$, and $0 \leqslant b \leqslant 1$. b has been designed to allow surface and/or bottom increase of the vertical resolution (Fig. 3.3.4).

3.3.5 z^*- or s^*-coordinate (add key_vvl)

This option is described in the Report by Levier *et al.* (2007), available on the *NEMO* web site.

3.3.6 level bathymetry and mask

Whatever the vertical coordinate used, the model offers the possibility of representing the bottom topography with steps that follow the face of the model cells (step like topography) [**?**]. The distribution of the steps in the horizontal is defined in a 2D integer array, mbathy, which gives the number of ocean levels ($i.e.$ those that are not masked) at each T-point. mbathy is computed from the meter bathymetry using the definiton of gdept as

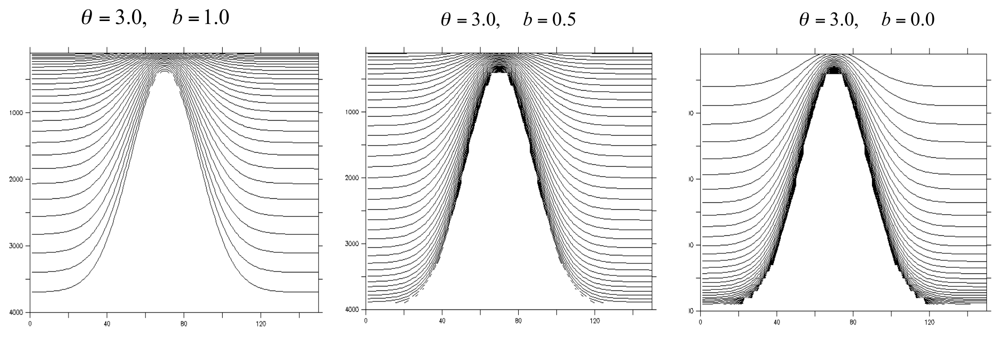


FIG. 3.7 – Examples of the stretching function applied to a sea mont ; from left to right : surface, surface and bottom, and bottom intensified resolutions

the number of T-points which gdept $\leq$ bathy. Note that in version NEMO v2.3, the user still has to provide the "level" bathymetry in a NetCDF file when using the full step option (*ln_zco*), rather than the bathymetry in meters : both will be allowed in future versions.

Modifications of the model bathymetry are performed in the *bat_ctl* routine (see *domzgr.F90* module) after mbathy is computed. Isolated grid points that do not communicate with another ocean point at the same level are eliminated.

In the case of the rigid-lid approximation when islands occur in the computational domain (*ln_dynspg_rl*=.true. and **key_island** is defined), the *mbathy* array must be provided and takes values from $-N$ to *jpk*-1. It provides the following information : $mbathy(i,j) = -n,\ n \in\ ]0,N]$, T-points are land points on the n^{th} island ; $mbathy(i,j) = 0$, T-points are land points on the main land (continent) ; $mbathy(i,j) = k$, the first k T- and w-points are ocean points, the others are points below the ocean floor.

This is used to compute the island barotropic stream function used in the rigid lid computation (see §10.7.4).

From the *mbathy* array, the mask fields are defined as follows :

$$
\begin{aligned}
tmask(i,j,k) &= \begin{cases} 1 & \text{if } k \leq mbathy(i,j) \\ 0 & \text{if } k \leq mbathy(i,j) \end{cases} \\
umask(i,j,k) &= \ tmask(i,j,k) \ * \ tmask(i+1,j,k) \\
vmask(i,j,k) &= \ tmask(i,j,k) \ * \ tmask(i,j+1,k) \\
fmask(i,j,k) &= \ tmask(i,j,k) \ * \ tmask(i+1,j,k) \\
&\quad * \, tmask(i,j,k) \ * \ tmask(i+1,j,k)
\end{aligned}
$$

Note that *wmask* is not defined as it is exactly equal to *tmask* with the numerical indexing used (§ 3.1.3). Moreover, the specification of closed lateral boundaries requires that at least the first and last rows and columns of the *mbathy* array are set to zero. In the particular case of an east-west cyclical boundary condition, *mbathy* has its last column equal to the second one and its first column equal to the last but one (and so too the mask arrays) (see § 7.2).

3.4 Time Discretisation

The time stepping used in *NEMO* is a three level scheme that can be represented as follows :

$$x^{t+\Delta t} = x^{t-\Delta t} + 2\,\Delta t\ \text{RHS}_x^{t-\Delta t,t,t+\Delta t} \tag{3.15}$$

where x stands for u, v, T or S ; RHS is the Right-Hand-Side of the corresponding time evolution equation ; Δt is the time step ; and the superscripts indicate the time at which a quantity is evaluated. Each term of the RHS is evaluated at a specific time step(s) depending on the physics with which it is associated.

The choice of the time step used for this evaluation is discussed below as well as the implications in terms of starting or restarting a model simulation. Note that the time stepping is generally performed in a one step operation. With such a complex and nonlinear system of equations it would be dangerous to let a prognostic variable evolve in time for each term separately.

The three level scheme requires three arrays for each prognostic variables. For each variable x there is x_b (before) and x_n (now). The third array, although referred to as x_a (after) in the code, is usually not the variable at the next time step ; but rather it is used to store the time derivative (RHS in (3.15)) prior to time-stepping the equation. Generally, the time stepping is performed once at each time step in *tranxt.F90* and *dynnxt.F90* modules, except for implicit vertical diffusion or sea surface height when time-splitting options are used.

3.4.1 Non-Diffusive Part — Leapfrog Scheme

The time stepping used for non-diffusive processes is the well-known leapfrog scheme. It is a time centred scheme, i.e. the RHS is evaluated at time step t, the now time step. It is only used for non-diffusive terms, that is momentum and tracer advection, pressure gradient, and Coriolis terms. This scheme is widely used for advective processes in low-viscosity fluids. It is an efficient method that achieves second-order accuracy with just one right hand side evaluation per time step. Moreover, it does not artificially damp linear oscillatory motion nor does it produce instability by amplifying the oscillations. These advantages are somewhat diminished by the large phase-speed error of the leapfrog scheme, and the unsuitability of leapfrog differencing for the representation of diffusive and Rayleigh damping processes. However, the most serious problem associated with the leapfrog scheme is a high-frequency computational noise called ”time-splitting” [**?**] that develops when the method is used to model non linear fluid dynamics : the even and odd time steps tend to diverge into a physical and a computational mode. Time splitting can be controlled through the use of an Asselin time filter (first designed by [**?**] and more comprehensively studied by **?**), or by periodically reinitialising the leapfrog solution through a single integration step with a two-level scheme. In *NEMO* we follow the first strategy :

$$x_F^t = x^t + \gamma \, \left[x_f^{t-\Delta t} - 2x^t + x^{t+\Delta t} \right] \tag{3.16}$$

where the subscript f denotes filtered values and γ is the Asselin coefficient. γ is initialized as *atfp* (namelist parameter). Its default value is *atfp*=0.1. This default value causes a significant dissipation of high frequency motions. Recommended values in idealized studies of shallow water turbulence are two orders of magnitude smaller ([**?**]). Both strategies do, nevertheless, degrade the accuracy of the calculation from second to first order. The leapfrog scheme combined with a Robert-Asselin time filter has been preferred to other time differencing schemes such as predictor corrector or trapezoidal schemes, because the user has an explicit and simple control of the magnitude of the time diffusion of the scheme. In association with the 2nd order centred space discretisation of the advective terms in the momentum and tracer equations, it avoids implicit numerical diffusion in both the time and space discretisations of the advective term : they are both set explicitly by the user through the Robert-Asselin filter parameter and the viscous and diffusive coefficients.

Alternative time stepping schemes are currently under investigation.

3.4.2 Diffusive Part — Forward or Backward Scheme

The leapfrog differencing scheme is unsuitable for the representation of diffusive and damping processes. For a tendancy D_x, representing a diffusive term or a restoring term to a tracer climatology (when present, see § 4.6), a forward time differencing scheme is used :

$$x^{t+\Delta t} = x^{t-\Delta t} + 2 \, \Delta t \; {D_x}^{t-\Delta t} \tag{3.17}$$

This is diffusive in time and conditionally stable. For example, the conditions for stability of second and fourth order horizontal diffusion schemes are [**?**] :

$$A^h < \begin{cases} \dfrac{e^2}{8 \; \Delta t} & \text{laplacian diffusion} \\ \dfrac{e^4}{64 \; \Delta t} & \text{bilaplacian diffusion} \end{cases} \tag{3.18}$$

where e is the smallest grid size in the two horizontal directions and A^h is the mixing coefficient. The linear constraint (3.18) is a necessary condition, but not sufficient. If it is not satisfied, even mildly, then the model soon becomes wildly unstable. The instability can be removed by either reducing the length of the time steps or reducing the mixing coefficient.

For the vertical diffusion terms, a forward time differencing scheme can be used, but usually the numerical stability condition implies a strong constraint on the time step. Two solutions are available in *NEMO* to overcome the stability constraint : (a) a forward time differencing scheme using a time splitting technique (*ln_zdfexp*=.true.) or (b) a backward (or implicit) time differencing scheme by *ln_zdfexp*=.false.). In (a), the master time step Δt is cut into N fractional time steps so that the stability criterion is reduced by a factor of N. The computation is done as follows :

$$\begin{aligned} &u_*^{t-\Delta t} = u^{t-\Delta t} \\ &u_*^{t-\Delta t+L\frac{2\Delta t}{N}} = u_*^{t-\Delta t+(L-1)\frac{2\Delta t}{N}} + \frac{2\Delta t}{N}\;\mathrm{DF}^{t-\Delta t+(L-1)\frac{2\Delta t}{N}} \quad \text{for } L = 1 \text{ to } N \\ &u^{t+\Delta t} = u_*^{t+\Delta t} \end{aligned} \tag{3.19}$$

with DF a vertical diffusion term. The number of fractional time steps, N, is given by setting *n_zdfexp*, (namelist parameter). The scheme (b) is unconditionally stable but diffusive. It can be written as follows :

$$x^{t+\Delta t} = x^{t-\Delta t} + 2\,\Delta t\;\mathrm{RHS}_x^{t+\Delta t} \tag{3.20}$$

This scheme is rather time consuming since it requires a matrix inversion, but it becomes attractive since a splitting factor of 3 or more is needed for the forward time differencing scheme. For example, the finite difference approximation of the temperature equation is :

$$\frac{T(k)^{t+1} - T(k)^{t-1}}{2\;\Delta t} \equiv \mathrm{RHS} + \frac{1}{e_{3T}}\delta_k\left[\frac{A_w^{vT}}{e_{3w}}\delta_{k+1/2}\left[T^{t+1}\right]\right] \tag{3.21}$$

where RHS is the right hand side of the equation except for the vertical diffusion term. We rewrite (3.20) as :

$$-c(k+1)\;u^{t+1}(k+1) + d(k)\;u^{t+1}(k) - \;c(k)\;u^{t+1}(k-1) \equiv b(k) \tag{3.22}$$

where

$$\begin{aligned} c(k) &= A_w^{vm}(k)\,/\,e_{3uw}(k) \\ d(k) &= e_{3u}(k)\,/\,(2\Delta t) + c_k + c_{k+1} \\ b(k) &= e_{3u}(k)\;\left(u^{t-1}(k)\,/\,(2\Delta t) + \mathrm{RHS}\right) \end{aligned}$$

(3.22) is a linear system of equations which associated matrix is tridiagonal. Moreover, $c(k)$ and $d(k)$ are positive and the diagonal term is greater than the sum of the two extra-diagonal terms, therefore a special adaptation of the Gauss elimination procedure is used to find the solution (see for example **?**).

3.4.3 Start/Restart strategy

```
!-----------------------------------------------------------------------
&namrun        !   parameters of the run
!-----------------------------------------------------------------------
   no          =       0   !  job number
   cexper      =  "ORCA2"  !  experience name
   ln_rstart   = .false.   !  start from rest (F) or from a restart file (T)
   nrstdt      =       0   !  restart control = 0 nit000 is not compared to the restart file value
                           !                  = 1 use ndate0 in namelist (not the value in the restart file)
                           !                  = 2 calendar parameters read in the restart file
   nit000      =       1   !  first time step
   nitend      =    5475   !  last  time step
   ndate0      =  010101   !  initial calendar date yymmdd (used if nrstdt=1)
   nleapy      =       0   !  Leap year calendar (1) or not (0)
   ninist      =       0   !  output the initial state (1) or not (0)
   nstock      =    5475   !  frequency of creation of a restart file
   nwrite      =    5475   !  frequency of write in the output file
   ln_dimgnnn  = .false.   !  DIMG file format: 1 file for all processors (F) or by processor (T)
/
```

The first time step of this three level scheme when starting from initial conditions is a forward step (Euler time integration) :

$$x^1 = x^0 + \Delta t \, \mathrm{RHS}^0 \tag{3.23}$$

It is also possible to restart from a previous computation, by using a restart file. The restart strategy is designed to ensure perfect restartability of the code : the user should obtain the same results to machine precision either by running the model for $2N$ time steps in one go, or by performing two consecutive experiments of N steps with a restart. This requires saving two time levels and many auxiliary data in the restart files in machine precision.

Note that when a semi-implicit scheme is used to evaluate the hydrostatic pressure gradient (see §5.3.4), an extra three-dimensional field has to be added in the restart file to ensure an exact restartability. This is done only optionally via the namelist parameter *nn_dynhpg_rst*, so that a reduction of the size of restart file can be obtained when the restartability is not a key issue (operational oceanography or ensemble simulation for seasonal forcast).

Ocean Tracers (TRA)

Contents

Using the representation described in Chap. 3, several semi-discrete space forms of the tracer equations are available depending on the vertical coordinate used and on the physics used. In all the equations presented here, the masking has been omitted for simplicity. One must be aware that all the quantities are masked fields and that each time a mean or difference operator is used, the resulting field is multiplied by a mask.

The two active tracers are potential temperature and salinity. Their prognostic equations can be summarized as follows :

$$\text{NXT} = \text{ADV} + \text{LDF} + \text{ZDF} + \text{SBC}\ (+\text{QSR})\ (+\text{BBC})\ (+\text{BBL})\ (+\text{DMP})$$

NXT stands for next, referring to the time-stepping. From left to right, the terms on the rhs of the tracer equations are the advection (ADV), the lateral diffusion (LDF), the vertical diffusion (ZDF), the contributions from the external forcings (SBC : Surface Boundary Condition, QSR : penetrative Solar Radiation, and BBC : Bottom Boundary Condition), the contribution from the bottom boundary Layer (BBL) parametrisation, and an internal damping (DMP) term. The terms QSR, BBC, BBL and DMP are optional. The external forcings and parameterisations require complex inputs and complex calculations (e.g. bulk formulae, estimation of mixing coefficients) that are carried out in the SBC, LDF and ZDF modules and described in chapters §6, §8 and §9, respectively. Note that *tranpc.F90,* the non-penetrative convection module, although (temporarily) located in the NEMO/OPA/TRA directory, is described with the model vertical physics (ZDF).

In the present chapter we also describe the diagnostic equations used to compute the sea-water properties (density, Brunt-Vaisälä frequency, specific heat and freezing point) although the associated modules ($i.e.$ *eosbn2.F90*, *ocfzpt.F90* and *phycst.F90*) are (temporarily) located in the NEMO/OPA directory.

The different options available to the user are managed by namelist logical or CPP keys. For each equation term *ttt*, the namelist logicals are *ln_trattt_xxx*, where *xxx* is a 3 or 4 letter acronym accounting for each optional scheme. The CPP key (when it exists) is **key_trattt**. The corresponding code can be found in the *trattt* or *trattt_xxx* module, in the NEMO/OPA/TRA directory.

The user has the option of extracting each tendency term on the rhs of the tracer equation for output (**key_trdtra** is defined), as described in Chap. 10.

4.1 Tracer Advection (*traadv.F90*)

```
!-----------------------------------------------------------------------
&nam_traadv    !   advection scheme for tracer
!-----------------------------------------------------------------------
   ln_traadv_cen2   =  .true.   !  2nd order centered scheme
   ln_traadv_tvd    =  .false.  !  TVD scheme
   ln_traadv_muscl  =  .false.  !  MUSCL scheme
   ln_traadv_muscl2 =  .false.  !  MUSCL2 scheme + cen2 at boundaries
   ln_traadv_ubs    =  .false.  !  UBS scheme
/
```

The advection tendency of a tracer in flux form is the divergence of the advective fluxes. Its discrete expression is given by :

$$ADV_\tau = -\frac{1}{b_T} \left(\delta_i \left[e_{2u}\, e_{3u}\; u\; \tau_u \right] + \delta_j \left[e_{1v}\, e_{3v}\; v\; \tau_v \right] \right) - \frac{1}{e_{3T}}\; \delta_k \left[w\; \tau_w \right] \tag{4.1}$$

where τ is either T or S, and $b_T = e_{1T}\, e_{2T}\, e_{3T}$ is the volume of T-cells. In pure z-coordinate (**key_zco** is defined), it reduces to :

$$ADV_\tau = -\frac{1}{e_{1T}\, e_{2T}} \left(\delta_i \left[e_{2u}\;\; u\; \tau_u \right] + \delta_j \left[e_{1v}\, v\; \tau_v \right] \right) - \frac{1}{e_{3T}} \delta_k \left[w\; \tau_w \right] \tag{4.2}$$

since the vertical scale factors are functions of k only, and thus $e_{3u} = e_{3v} = e_{3T}$. The flux form in (4.1) requires implicitly the use of the continuity equation. Indeed, it is obtained by using the following equality : $\nabla \cdot (\mathbf{U}\,\mathrm{T}) = \mathbf{U} \cdot \nabla \mathrm{T}$ which results from the use of the continuity equation, $\nabla \cdot \mathbf{U} = 0$ or $\partial_t e_3 + e_3\;\nabla \cdot \mathbf{U} = 0$ in constant (default option) or variable (**key_vvl** defined) volume case, respectively. Therefore it is of paramount importance to design the discrete analogue of the advection tendency so that it is consistent with the continuity equation in order to enforce the conservation properties of the continuous equations. In other words, by substituting τ by 1 in (4.1) we recover the discrete form of the continuity equation which is used to calculate the vertical velocity.

The key difference between the advection schemes used in *NEMO* is the choice made in space and time interpolation to define the value of the tracer at the velocity points (Fig. 4.1).

Along solid lateral and bottom boundaries a zero tracer flux is naturally specified, since the normal velocity is zero there. At the sea surface the boundary condition depends on the type of sea surface chosen :

rigid-lid formulation : $w = 0$ at the surface, so the advective fluxes through the surface are zero.

linear free surface : the first level thickness is constant in time : the vertical boundary condition is applied at the fixed surface $z = 0$ rather than on the moving surface $z = \eta$. There is a non-zero advective flux which is set for all advection schemes as the product of surface velocity (at $z = 0$) by the first level tracer value : $\tau_w|_{k=1/2} = T_{k=1}$.

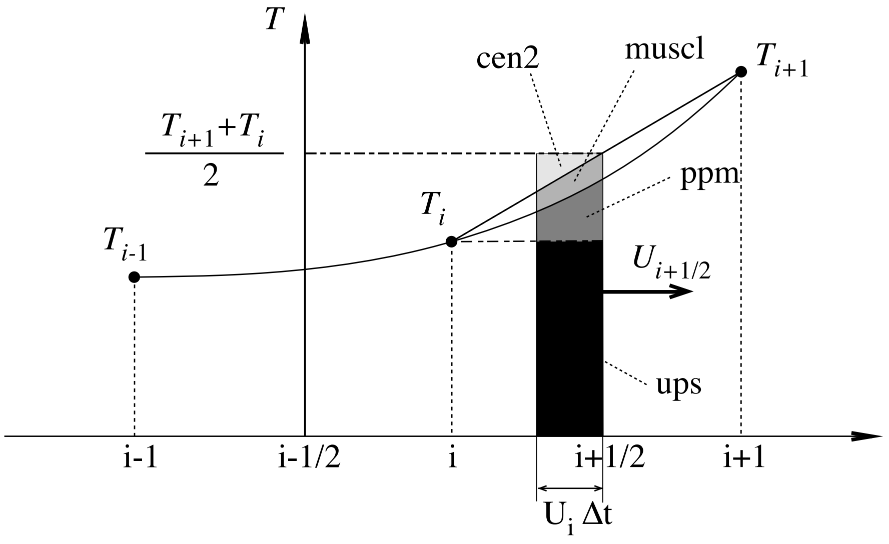


FIG. 4.1 – Schematic representation of some ways used to evaluate the tracer value at u-point and the amount of tracer exchanged between two neighbouring grid points. Upsteam biased scheme (ups) : the upstream value is used and the black area is exchanged. Piecewise parabolic method (ppm) : a parabolic interpolation is used and the black and dark grey areas are exchanged. Monotonic upstream scheme for conservative laws (muscl) : a parabolic interpolation is used and black, dark grey and grey areas are exchanged. Second order scheme (cen2) : the mean value is used and black, dark grey, grey and light grey areas are exchanged. Note that this illustration does not include the flux limiter used in ppm and muscl schemes.

non-linear free surface : (**key_vvl** is defined) convergence/divergence in the first ocean level moves the free surface up/down. There is no tracer advection through it so that the advective fluxes through the surface are also zero

In all cases, this boundary condition retains local conservation of tracer. Global conservation is obtained in both rigid-lid and non-linear free surface cases, but not in the linear free surface case. Nevertheless, in the latter case, it is achieved to a good approximation since the non-conservative term is the product of the time derivative of the tracer and the free surface height, two quantities that are not correlated (see §2.2.2, and also **???**).

The velocity field that appears in (4.1) and (4.2) is the centred (*now*) *eulerian* ocean velocity (see Chap. 5). When advective bottom boundary layer (*bbl*) and/or eddy induced velocity (*eiv*) parameterisations are used it is the *now effective* velocity ($i.e.$ the sum of the eulerian, the bbl and/or the eiv velocities) which is used.

The choice of an advection scheme is made in the *nam_traadv* namelist, by setting to *true* one and only one of the logicals *ln_traadv_xxx*. The corresponding code can be found in the *traadv_xxx.F90* module, where *xxx* is a 3 or 4 letter acronym corresponding to each scheme. Details of the advection schemes are given below. The choice of an advection scheme is a complex matter which depends on the model physics, model resolution, type of tracer, as well as the issue of numerical cost.

Note that (1) cen2, cen4 and TVD schemes require an explicit diffusion operator while the other schemes are diffusive enough so that they do not require additional diffusion ; (2) cen2, cen4, MUSCL2, and UBS are not *positive* schemes [1] , implying that false extrema are permitted. Their use is not recommended on passive tracers ; (3) It is highly recommended that the same advection-diffusion scheme is used on both active and passive tracers. Indeed, if a source or sink of a passive tracer depends on an active one, the difference of treatment of active and passive tracers can create very nice-looking frontal structures that are pure numerical artefacts.

4.1.1 2^{nd} order centred scheme (cen2) (*ln_traadv_cen2*=.true.)

In the centred second order formulation, the tracer at velocity points is evaluated as the mean of the two neighbouring T-point values. For example, in the i-direction :

$$\tau_u^{cen2} = \overline{T}^{\,i+1/2} \tag{4.3}$$

The scheme is non diffusive ($i.e.$ it conserves the tracer variance, τ^2) but dispersive ($i.e.$ it may create false extrema). It is therefore notoriously noisy and must be used in conjunction with an explicit diffusion operator to produce a sensible solution. The associated time-stepping is performed using a leapfrog scheme in conjunction with an Asselin time-filter, so T in (4.3) is the *now* tracer value. The centered second order advection is computed in the *traadv_cen2.F90* module. In this module, it is also proposed to combine the *cen2* scheme with an upstream scheme in specific areas which requires a strong diffusion in order to avoid the generation of false extrema. These areas are the vicinity of large river mouths, some straits with coarse resolution, and the vicinity of ice cover area ($i.e.$ when the ocean temperature is close to the freezing point).

Note that using the cen2 scheme, the overall tracer advection is of second order accuracy since both (4.1) and (4.3) have this order of accuracy. Note also that

4.1.2 4^{nd} order centred scheme (cen4) (*ln_traadv_cen4*=.true.)

In the 4^{th} order formulation (to be implemented), tracer values are evaluated at velocity points as a 4^{th} order interpolation, and thus uses the four neighbouring T-points. For example, in the i-direction :

$$\tau_u^{cen4} = \overline{T - \frac{1}{6}\,\delta_i\left[\delta_{i+1/2}[T]\,\right]}^{\,i+1/2} \tag{4.4}$$

[1]negative values can appear in an initially strictly positive tracer field which is advected

Strictly speaking, the cen4 scheme is not a 4^{th} order advection scheme but a 4^{th} order evaluation of advective fluxes, since the divergence of advective fluxes (4.1) is kept at 2^{nd} order. The phrase "4^{th} order scheme" used in oceanographic literature is usually associated with the scheme presented here. Introducing a *true* 4^{th} order advection scheme is feasible but, for consistency reasons, it requires changes in the discretisation of the tracer advection together with changes in both the continuity equation and the momentum advection terms.

A direct consequence of the pseudo-fourth order nature of the scheme is that it is not non-diffusive, i.e. the global variance of a tracer is not preserved using *cen4*. Furthermore, it must be used in conjunction with an explicit diffusion operator to produce a sensible solution. The time-stepping is also performed using a leapfrog scheme in conjunction with an Asselin time-filter, so T in (4.4) is the *now* tracer.

At a T-grid cell adjacent to a boundary (coastline, bottom and surface), an additional hypothesis must be made to evaluate τ_u^{cen4}. This hypothesis usually reduces the order of the scheme. Here we choose to set the gradient of T across the boundary to zero. Alternative conditions can be specified, such as a reduction to a second order scheme for these near boundary grid points.

4.1.3 Total Variance Dissipation scheme (TVD) (*ln_traadv_tvd*=.true.)

In the Total Variance Dissipation (TVD) formulation, the tracer at velocity points is evaluated using a combination of an upstream and a centred scheme. For example, in the i-direction :

$$\begin{aligned} \tau_u^{ups} &= \begin{cases} T_{i+1} & \text{if } \ u_{i+1/2} < 0 \\ T_i & \text{if } \ u_{i+1/2} \geq 0 \end{cases} \\ \tau_u^{tvd} &= \tau_u^{ups} + c_u \ \left(\tau_u^{cen2} - \tau_u^{ups} \right) \end{aligned} \tag{4.5}$$

where c_u is a flux limiter function taking values between 0 and 1. There exist many ways to define c_u, each correcponding to a different total variance decreasing scheme. The one chosen in *NEMO* is described in **?**. c_u only departs from 1 when the advective term produces a local extremum in the tracer field. The resulting scheme is quite expensive but *positive*. It can be used on both active and passive tracers. This scheme is tested and compared with MUSCL and the MPDATA scheme in **?** ; note that in this paper it is referred to as "FCT" (Flux corrected transport) rather than TVD. The TVD scheme is computed in the *traadv_tvd.F90* module.

For stability reasons (see §3.4), in (4.5) τ_u^{cen2} is evaluated using the *now* tracer while τ_u^{ups} is evaluated using the *before* tracer. In other words, the advective part of the scheme is time stepped with a leap-frog scheme while a forward scheme is used for the diffusive part.

4.1.4 Monotone Upstream Scheme for Conservative Laws (MUSCL) (*ln_traadv_muscl*=T)

The Monotone Upstream Scheme for Conservative Laws (MUSCL) has been implemented by **?**. In its formulation, the tracer at velocity points is evaluated assuming a linear tracer variation between two T-points (Fig.4.1). For example, in the i-direction :

$$\tau_u^{mus} = \begin{cases} \tau_i & + \frac{1}{2} \left(1 - \frac{u_{i+1/2}\,\Delta t}{e_{1u}} \right) \widetilde{\partial_i \tau} & \text{if } u_{i+1/2} \geqslant 0 \\ \tau_{i+1/2} & + \frac{1}{2} \left(1 + \frac{u_{i+1/2}\,\Delta t}{e_{1u}} \right) \widetilde{\partial_{i+1/2} \tau} & \text{if } u_{i+1/2} < 0 \end{cases} \tag{4.6}$$

where $\widetilde{\partial_i \tau}$ is the slope of the tracer on which a limitation is imposed to ensure the *positive* character of the scheme.

The time stepping is performed using a forward scheme, that is the *before* tracer field is used to evaluate τ_u^{mus}.

For an ocean grid point adjacent to land and where the ocean velocity is directed toward land, two choices are available : an upstream flux (*ln_traadv_muscl*=.true.) or a second order flux (*ln_traadv_muscl2*=.true.). Note that the latter choice does not ensure the *positive* character of the scheme. Only the former can be used on both active and passive tracers. The two MUSCL schemes are computed in the *traadv_tvd.F90* and *traadv_tvd2.F90* modules.

4.1.5 Upstream-Biased Scheme (UBS) (*ln_traadv_ubs*=.true.)

The UBS advection scheme is an upstream-biased third order scheme based on an upstream-biased parabolic interpolation. It is also known as the Cell Averaged QUICK scheme (Quadratic Upstream Interpolation for Convective Kinematics). For example, in the i-direction :

$$\tau_u^{ubs} = \overline{T}^{i+1/2} - \frac{1}{6} \begin{cases} \tau"_i & \text{if } u_{i+1/2} \geqslant 0 \\ \tau"_{i+1} & \text{if } u_{i+1/2} < 0 \end{cases} \tag{4.7}$$

where $\tau"_i = \delta_i \left[\delta_{i+1/2} \left[\tau \right] \right]$.

This results in a dissipatively dominant (i.e. hyper-diffusive) truncation error [**?**]. The overall performance of the advection scheme is similar to that reported in **?**. It is a relatively good compromise between accuracy and smoothness. It is not a *positive* scheme, meaning that false extrema are permitted, but the amplitude of such are significantly reduced over the centred second order method. Nevertheless it is not recommended that it should be applied to a passive tracer that requires positivity.

The intrinsic diffusion of UBS makes its use risky in the vertical direction where the control of artificial diapycnal fluxes is of paramount importance. Therefore the vertical flux is evaluated using the TVD scheme when *ln_traadv_ubs*=.true..

For stability reasons (see §3.4), in (4.7), the first term (which corresponds to a second order centred scheme) is evaluated using the *now* tracer (centred in time) while the second term (which is the diffusive part of the scheme), is evaluated using the *before* tracer (forward in time). This choice is discussed by **?** in the context of the QUICK advection scheme. UBS and QUICK schemes only differ by one coefficient. Replacing 1/6 with 1/8 in (4.7) leads to the QUICK advection scheme [**?**]. This option is not available through a namelist parameter, since the 1/6 coefficient is hard coded. Nevertheless it is quite easy to make the substitution in the *traadv_ubs.F90* module and obtain a QUICK scheme.

Note that :

(1) When a high vertical resolution $O(1m)$ is used, the model stability can be controlled by vertical advection (not vertical diffusion which is usually solved using an implicit scheme). Computer time can be saved by using a time-splitting technique on vertical advection. Such a technique has been implemented and validated in ORCA05 with 301 levels. It is not available in the current reference version.

(2) In a forthcoming release four options will be available for the vertical component used in the UBS scheme. τ_w^{ubs} will be evaluated using either *(a)* a centred 2^{nd} order scheme, or *(b)* a TVD scheme, or *(c)* an interpolation based on conservative parabolic splines following the **?** implementation of UBS in ROMS, or *(d)* a UBS. The 3^{rd} case has dispersion properties similar to an eighth-order accurate conventional scheme.

(3) It is straightforward to rewrite (4.7) as follows :

$$\tau_u^{ubs} = \tau_u^{cen4} + \frac{1}{12} \begin{cases} +\tau"_i & \text{if } u_{i+1/2} \geqslant 0 \\ -\tau"_{i+1} & \text{if } u_{i+1/2} < 0 \end{cases} \tag{4.8}$$

or equivalently

$$u_{i+1/2}\,\tau_u^{ubs} = u_{i+1/2}\,\overline{T - \frac{1}{6}\,\delta_i\left[\delta_{i+1/2}[T]\,\right]}^{\,i+1/2} - \frac{1}{2}|u|_{i+1/2}\,\frac{1}{6}\,\delta_{i+1/2}[\tau"_i] \tag{4.9}$$

(4.8) has several advantages. Firstly, it clearly reveals that the UBS scheme is based on the fourth order scheme to which an upstream-biased diffusion term is added. Secondly, this emphasises that the 4^{th} order part (as well as the 2^{nd} order part as stated above) has to be evaluated at the *now* time step using (4.7). Thirdly, the diffusion term is in fact a biharmonic operator with an eddy coefficient which is simply proportional to the velocity : $A_u^{lm} = -\frac{1}{12}\,{e_{1u}}^3\,|u|$. Note that NEMO v2.3 still uses (4.7), not (4.8). This should be changed in forthcoming release.

4.1.6 QUICKEST scheme (QCK) (*ln_traadv_qck*=.true.)

The Quadratic Upstream Interpolation for Convective Kinematics with Estimated Streaming Terms (QUICKEST) scheme proposed by **?** is the third order Godunov scheme. It is associated with the ULTIMATE QUICKEST limiter [**?**]. It has been implemented in NEMO by G. Reffray (MERCATOR-ocean) and can be found in the *traadv_qck.F90* module. The resulting scheme is quite expensive but *positive*. It can be used on both active and passive tracers. Nevertheless, the intrinsic diffusion of QCK makes its use risky in the vertical direction where the control of artificial diapycnal fluxes is of paramount importance. Therefore the vertical flux is evaluated using the CEN2 scheme. This no more ensure the positivity of the scheme. The use of TVD in the vertical direction as for the UBS case should be implemented to maintain the property.

4.1.7 Piecewise Parabolic Method (PPM) (*ln_traadv_ppm*=.true.)

The Piecewise Parabolic Method (PPM) proposed by Colella and Woodward (1984) is based on a quadradic piecewise rebuilding. Like the QCK scheme, it is associated with the ULTIMATE QUICKEST limiter [**?**]. It has been implemented in *NEMO* by G. Reffray (MERCATOR-ocean) but is not yet offered in the reference version 3.0.

4.2 Tracer Lateral Diffusion (*traldf.F90*)

```
!-----------------------------------------------------------------------
&nam_traldf    !   lateral diffusion scheme for tracer
!-----------------------------------------------------------------------
!                                !  Type of the operator :
   ln_traldf_lap    =  .true.    !     laplacian operator
   ln_traldf_bilap  =  .false.   !     bilaplacian operator
                                 !  Direction of action  :
   ln_traldf_level  =  .false.   !     iso-level
   ln_traldf_hor    =  .false.   !     horizontal (geopotential)         (require "key_ldfslp" when ln_sco=T)
   ln_traldf_iso    =  .true.    !     iso-neutral                       (require "key_ldfslp")
!                                !  Coefficient
   aht0        =  2000.          !     horizontal eddy diffusivity for tracers [m2/s]
   ahtb0       =     0.          !     background eddy diffusivity for ldf_iso [m2/s]
   aeiv0       =  2000.          !     eddy induced velocity coefficient [m2/s]    (require "key_traldf_eiv")
/
```

The options available for lateral diffusion are a laplacian (rotated or not) or a biharmonic operator, the latter being more scale-selective (more diffusive at small scales). The specification of eddy diffusivity coefficients (either constant or variable in space and time) as well as the computation of the slope along which the operators act, are performed in the *ldftra.F90* and *ldfslp.F90* modules, respectively. This is described in Chap. 8. The lateral diffusion of tracers is evaluated using a forward scheme, $i.e.$ the tracers appearing in its expression are the *before* tracers in time, except for the pure vertical component that appears when a rotation tensor is used. This latter term is solved implicitly together with the vertical diffusion term (see §3.4).

4.2.1 Iso-level laplacian operator (lap) (*ln_traldf_lap*=.true.)

A laplacian diffusion operator ($i.e.$ a harmonic operator) acting along the model surfaces is given by :

$$D_T^{lT} = \frac{1}{b_T} \left(\delta_i \left[A_u^{lT} \; \frac{e_{2u}\, e_{3u}}{e_{1u}} \; \delta_{i+1/2}[T] \right] + \delta_j \left[A_v^{lT} \; \frac{e_{1v}\, e_{3v}}{e_{2v}} \; \delta_{j+1/2}[T] \right] \right) \tag{4.10}$$

where $b_T = e_{1T}\, e_{2T}\, e_{3T}$ is the volume of T-cells. It can be found in the *traadv_lap.F90* module.

This lateral operator is computed in *traldf_lap.F90*. It is a *horizontal* operator ($i.e.$ acting along geopotential surfaces) in the z-coordinate with or without partial step, but is simply an iso-level operator in the s-coordinate. It is thus used when, in addition to *ln_traldf_lap*=.true., we have *ln_traldf_level*=.true., or *ln_traldf_hor*=*ln_zco*=.true.. In both cases, it significantly contributes to diapycnal mixing. It is therefore not recommended.

Note that (a) In pure z-coordinate (**key_zco** is defined), $e_{3u} = e_{3v} = e_{3T}$, so that the vertical scale factors disappear from (4.10) ; (b) In partial step z-coordinate (*ln_zps*=.true.), tracers in horizontally adjacent cells are located at different depths in the vicinity of the bottom. In this case, horizontal derivatives in (4.10) at the bottom level require a specific treatment. They are calculated in the *zpshde.F90* module, described in §4.9.

4.2.2 Rotated laplacian operator (iso) (*ln_traldf_lap*=.true.)

The general form of the second order lateral tracer subgrid scale physics (2.43) takes the following semi-discrete space form in z- and s-coordinates :

$$\begin{aligned} D_T^{lT} = \frac{1}{b_T} \Bigg\{ \; & \delta_i \left[A_u^{lT} \left(\frac{e_{2u}\, e_{3u}}{e_{1u}} \, \delta_{i+1/2}[T] - e_{2u}\, r_{1u} \, \overline{\overline{\delta_{k+1/2}[T]}}^{\,i+1/2,k} \right) \right] \\ & + \delta_j \left[A_v^{lT} \left(\frac{e_{1v}\, e_{3v}}{e_{2v}} \, \delta_{j+1/2}[T] - e_{1v}\, r_{2v} \, \overline{\overline{\delta_{k+1/2}[T]}}^{\,j+1/2,k} \right) \right] \\ & + \delta_k \left[A_w^{lT} \left(- e_{2w}\, r_{1w} \, \overline{\overline{\delta_{i+1/2}[T]}}^{\,i,k+1/2} \right. \right. \\ & \qquad\qquad - e_{1w}\, r_{2w} \, \overline{\overline{\delta_{j+1/2}[T]}}^{\,j,k+1/2} \\ & \qquad\qquad \left. \left. + \frac{e_{1w}\, e_{2w}}{e_{3w}} \left(r_{1w}^2 + r_{2w}^2 \right) \delta_{k+1/2}[T] \right) \right] \quad \Bigg\} \end{aligned} \tag{4.11}$$

where $b_T = e_{1T}\, e_{2T}\, e_{3T}$ is the volume of T-cells, r_1 and r_2 are the slopes between the surface of computation (z- or s-surfaces) and the surface along which the diffusion operator acts ($i.e.$ horizontal or iso-neutral surfaces). It is thus used when, in addition to *ln_traldf_lap*= .true., we have *ln_traldf_iso*=.true., or both *ln_traldf_hor*=.true. and *ln_zco*=.true.. The way these slopes are evaluated is given in §8.2. At the surface, bottom and lateral boundaries, the turbulent fluxes of heat and salt are set to zero using the mask technique (see §7.1).

The operator in (4.11) involves both lateral and vertical derivatives. For numerical stability, the vertical second derivative must be solved using the same implicit time scheme as that used in the vertical physics (see §4.3). For computer efficiency reasons, this term is not computed in the *traldf_iso.F90* module, but in the *trazdf.F90* module where, if iso-neutral mixing is used, the vertical mixing coefficient is simply increased by $\frac{e_{1w}\, e_{2w}}{e_{3w}} \left(r_{1w}^2 + r_{2w}^2 \right)$.

This formulation conserves the tracer but does not ensure the decrease of the tracer variance. Nevertheless the treatment performed on the slopes (see §8) allows the model to run safely without any additional background horizontal diffusion [**?**]. An alternative scheme developed by **?** which preserves both tracer and its variance is currently been tested in *NEMO* . It should be available in a forthcoming release.

Note that in the partial step z-coordinate (*ln_zps*=.true.), the horizontal derivatives at the bottom level in (4.11) require a specific treatment. They are calculated in module zpshde, described in §4.9.

4.2.3 Iso-level bilaplacian operator (bilap) (*ln_traldf_bilap*=.true.)

The lateral fourth order bilaplacian operator on tracers is obtained by applying (4.10) twice. It requires an additional assumption on boundary conditions : the first and third derivative terms normal to the coast are set to zero. It is used when, in addition to *ln_traldf_bilap*=.true., we have *ln_traldf_level*=.true., or both *ln_traldf_hor*=.true. and *ln_zco*=.false.. In both cases, it can contribute diapycnal mixing, although less than in the laplacian case. It is therefore not recommended.

Note that in the code, the bilaplacian routine does not call the laplacian routine twice but is rather a separate routine that can be found in the *traldf_bilap.F90* module. This is due to the fact that we introduce the eddy diffusivity coefficient, A, in the operator as : $\nabla \cdot \nabla \left(A \nabla \cdot \nabla T \right)$, instead of $-\nabla \cdot a \nabla \left(\nabla \cdot a \nabla T \right)$ where $a = \sqrt{|A|}$ and $A < 0$. This was a mistake : both formulations ensure the total variance decrease, but the former requires a larger number of code-lines. It will be corrected in a forthcoming release.

4.2.4 Rotated bilaplacian operator (bilapg) (*ln_traldf_bilap*=.true.)

The lateral fourth order operator formulation on tracers is obtained by applying (4.11) twice. It requires an additional assumption on boundary conditions : first and third derivative terms normal to the coast, the bottom and the surface are set to zero. It can be found in the *traldf_bilapg.F90*.

It is used when, in addition to *ln_traldf_bilap*=.true., we have *ln_traldf_iso*= .true, or both *ln_traldf_hor*=.true. and *ln_zco*=.true.. Nevertheless, this rotated bilaplacian operator has never been seriously tested. No warranties that it is neither free of bugs or correctly formulated. Moreover, the stability range of such an operator will be probably quite narrow, requiring a significantly smaller time-step than the one used on unrotated operator.

4.3 Tracer Vertical Diffusion (*trazdf.F90*)

```
!-----------------------------------------------------------------------
&namzdf        !   vertical physics
!-----------------------------------------------------------------------
   avm0        =   1.2e-4  !  vertical eddy viscosity   [m2/s]          (background Kz if not "key_zdfcst")
   avt0        =   1.2e-5  !  vertical eddy diffusivity [m2/s]          (background Kz if not "key_zdfcst")
   ln_zdfnpc   = .false.   !  convection: Non-Penetrative algorithm (T) or not (F)
   ln_zdfevd   = .true.    !  convection: enhanced vertical diffusion (T) or not (F)
   avevd       = 100.      !  vertical coefficient for enhanced diffusion scheme [m2/s]
   n_evdm      =   1       !  enhanced mixing apply on tracer (=0) or on tracer and momentum (=1)
   ln_zdfexp   =  .false.  !  split explicit (T) or implicit (F) time stepping
   n_zdfexp    =   3       !  number of sub-timestep for ln_zdfexp=T
/
```

The formulation of the vertical subgrid scale tracer physics is the same for all the vertical coordinates, and is based on a laplacian operator. The vertical diffusion operator given by (2.43) takes the following semi-discrete space form :

$$
\begin{aligned}
D_T^{vT} &= \frac{1}{e_{3T}}\ \delta_k \left[\frac{A_w^{vT}}{e_{3w}} \delta_{k+1/2}[T] \right] \\
D_T^{vS} &= \frac{1}{e_{3T}}\ \delta_k \left[\frac{A_w^{vS}}{e_{3w}} \delta_{k+1/2}[S] \right]
\end{aligned}
\tag{4.12}
$$

where A_w^{vT} and A_w^{vS} are the vertical eddy diffusivity coefficients on temperature and salinity, respectively. Generally, $A_w^{vT} = A_w^{vS}$ except when double diffusive mixing is parameterised ($i.e.$ **key_zdfddm** is defined). The way these coefficients are evaluated is given in §9 (ZDF). Furthermore, when iso-neutral mixing is used, both mixing coefficients are increased by $\frac{e_{1w}\,e_{2w}}{e_{3w}}\ \left(r_{1w}^2 + r_{2w}^2\right)$ to account for the vertical second derivative of (4.11).

At the surface and bottom boundaries, the turbulent fluxes of heat and salt must be specified. At the surface they are prescribed from the surface forcing and added in a dedicated routine (see §4.4.1), whilst at the bottom they are set to zero for heat and salt unless a geothermal flux forcing is prescribed as a bottom boundary condition (see §4.4.3).

The large eddy coefficient found in the mixed layer together with high vertical resolution implies that in the case of explicit time stepping (*ln_zdfexp*=.true.) there would be too restrictive a constraint on the time step. Therefore, the default implicit time stepping is preferred for the vertical diffusion since it overcomes the stability constraint. A forward time differencing scheme (*ln_zdfexp*=.true.) using a time splitting technique (*n_zdfexp* > 1) is provided as an alternative. Namelist variables *ln_zdfexp* and *n_zdfexp* apply to both tracers and dynamics.

4.4 External Forcing

4.4.1 Surface boundary condition (*trasbc.F90*)

The surface boundary condition for tracers is implemented in a separate module (*trasbc.F90*) instead of entering as a boundary condition on the vertical diffusion operator (as in the case of momentum). This has been found to enhance readability of the code. The two formulations are completely equivalent ; the forcing terms in trasbc are the surface fluxes divided by the thickness of the top model layer. Following **?** the forcing on an ocean tracer, c, can be split into two parts : F_{ext}, the flux of tracer crossing the sea surface and not linked with the water exchange with the atmosphere, F_{wf}^d, and F_{wf}^i the forcing on the tracer concentration associated with this water exchange. The latter forcing has two components : a direct effect of change in concentration associated with the tracer carried by the water flux, and an indirect concentration/dilution effect :

$$
\begin{aligned}
F^C &= F_{ext} + F_{wf}^d + F_{wf}^i \\
&= F_{ext} - (c_E\,E - c_p\,P - c_R\,R) + c\,(E - P - R)
\end{aligned}
$$

Two cases must be distinguished, the nonlinear free surface case (**key_vvl** is defined) and the linear free surface case. The first case is simpler, because the indirect concentration/dilution effect is naturally taken into account by letting the vertical scale factors vary in time. The salinity of water exchanged at the surface is assumed to be zero, so there is no salt flux at the free surface, except in the presence of sea ice. The heat flux at the free surface is the sum of F_{ext}, the direct heating/cooling (by the total non-penetrative heat flux) and F_{wf}^{e} the heat carried by the water exchanged through the surface (evaporation, precipitation, runoff). The temperature of precipitation is not well known. In the model we assume that this water has the same temperature as the sea surface temperature. The resulting forcing terms for temperature T and salinity S are :

$$
\begin{aligned}
F^T &= \frac{Q_{ns}}{\rho_o \; C_p \, e_{3T}} - \frac{\text{EMP} \;\; T|_{k=1}}{e_{3T}} \\
F^S &= \frac{\text{EMP}_S \;\; S|_{k=1}}{e_{3T}}
\end{aligned}
\tag{4.13}
$$

where EMP is the freshwater budget (evaporation minus precipitation minus river runoff) which forces the ocean volume, Q_{ns} is the non-penetrative part of the net surface heat flux (difference between the total surface heat flux and the fraction of the short wave flux that penetrates into the water column), the product $\text{EMP}_S \; . \; S|_{k=1}$ is the ice-ocean salt flux, and $S|_{k=1}$ is the sea surface salinity (*SSS*). The total salt content is conserved in this formulation (except for the effect of the Asselin filter).

In the second case (linear free surface), the vertical scale factors are fixed in time so that the concentration/dilution effect must be added in the *trasbc.F90* module. Because of the hypothesis made for the temperature of precipitation and runoffs, $F_{wf}^{e} + F_{wf}^{i} = 0$ for temperature. The resulting forcing term for temperature is :

$$
F^T = \frac{Q_{ns}}{\rho_o \; C_p \, e_{3T}} \tag{4.14}
$$

The salinity forcing is still given by (4.13) but the definition of EMP_S is different : it is the total surface freshwater budget (evaporation minus precipitation minus river runoff plus the rate of change of the sea ice thickness). The total salt content is not exactly conserved (**?**. See also §2.2.2).

In the case of the rigid lid approximation, the surface salinity forcing F^s is also expressed by (4.13), but now the global integral of the product of EMP and S, is not compensated by the advection of fluid through the top level : this is because in the rigid lid case *w(k=1) = 0* (in contrast to the linear free surface case). As a result, even if the budget of *EMP* is zero on average over the whole ocean domain, the associated salt flux is not, since sea-surface salinity and *EMP* are intrinsically correlated (high *SSS* are found where evaporation is strong whilst low *SSS* is usually associated with high precipitation or river runoff).

The Q_{ns} and *EMP* fields are defined and updated in the *sbcmod.F90* module (see §6).

4.4.2 Solar Radiation Penetration (*traqsr.F90*)

```
!----------------------------------------------------------------------
```

```
&namqsr        !   penetrative solar radiation
!-----------------------------------------------------------------------
   ln_traqsr   = .true.    !  penetrative solar radiation (T) or not (F)
   rabs        =   0.58    !  fraction of qsr associated with xsi1
   xsi1        =   0.35    !  first depth of extinction
   xsi2        =   23.0    !  second depth of extinction
/
```

When the penetrative solar radiation option is used (*ln_flxqsr*=.true.), the solar radiation penetrates the top few meters of the ocean, otherwise all the heat flux is absorbed in the first ocean level (*ln_flxqsr*=.false.). Thus, in the former case a term is added to the time evolution equation of temperature (2.1d) whilst the surface boundary condition is modified to take into account only the non-penetrative part of the surface heat flux :

$$\begin{aligned} \frac{\partial T}{\partial t} &= \ldots + \frac{1}{\rho_o\, C_p\, e_3}\, \frac{\partial I}{\partial k} \\ Q_{ns} &= Q_{\text{Total}} - Q_{sr} \end{aligned} \tag{4.15}$$

where I is the downward irradiance. The additional term in (4.15) is discretized as follows :

$$\frac{1}{\rho_o\, C_p\, e_3}\, \frac{\partial I}{\partial k} \equiv \frac{1}{\rho_o\, C_p\, e_{3T}} \delta_k \left[I_w \right] \tag{4.16}$$

A formulation involving two extinction coefficients is assumed for the downward irradiance I [**?**] :

$$I(z) = Q_{sr} \left[Re^{-z / \xi_1} + (1-R)\, e^{-z / \xi_2} \right] \tag{4.17}$$

where Q_{sr} is the penetrative part of the surface heat flux, ξ_1 and ξ_2 are two extinction length scales and R determines the relative contribution of the two terms. The default values used correspond to a Type I water in Jerlov's [1968] Jerlov reference to be added classification : $\xi_1 = 0.35\ m$, $\xi_2 = 23\ m$ and $R = 0.58$ (corresponding to *xsi1*, *xsi2* and *rabs* namelist parameters, respectively). I is masked (no flux through the ocean bottom), so all the solar radiation that reaches the last ocean level is absorbed in that level. The trend in (4.16) associated with the penetration of the solar radiation is added to the temperature trend, and the surface heat flux is modified in routine *traqsr.F90*. Note that in the z-coordinate, the depth of $T-$levels depends on the single variable k. A one dimensional array of the coefficients $gdsr(k) = Re^{-z_w(k)/\xi_1} + (1-R)e^{-z_w(k)/\xi_2}$ can then be computed once and saved in memory. Moreover *nksr*, the level at which $gdrs$ becomes negligible (less than the computer precision) is computed once, and the trend associated with the penetration of the solar radiation is only added until that level. Finally, note that when the ocean is shallow (¡ 200 m), part of the solar radiation can reach the ocean floor. In this case, we have chosen that all remaining radiation is absorbed in the last ocean level ($i.e.$ I_w is masked).

When coupling with a biological model (for example PISCES or LOBSTER), it is possible to calculate the light attenuation using information from the biology model. Without biological model, it is still possible to introduce a horizontal variation of the light attenuation by using the observed ocean surface color. At the time of writing, the latter has not been implemented in the reference version.

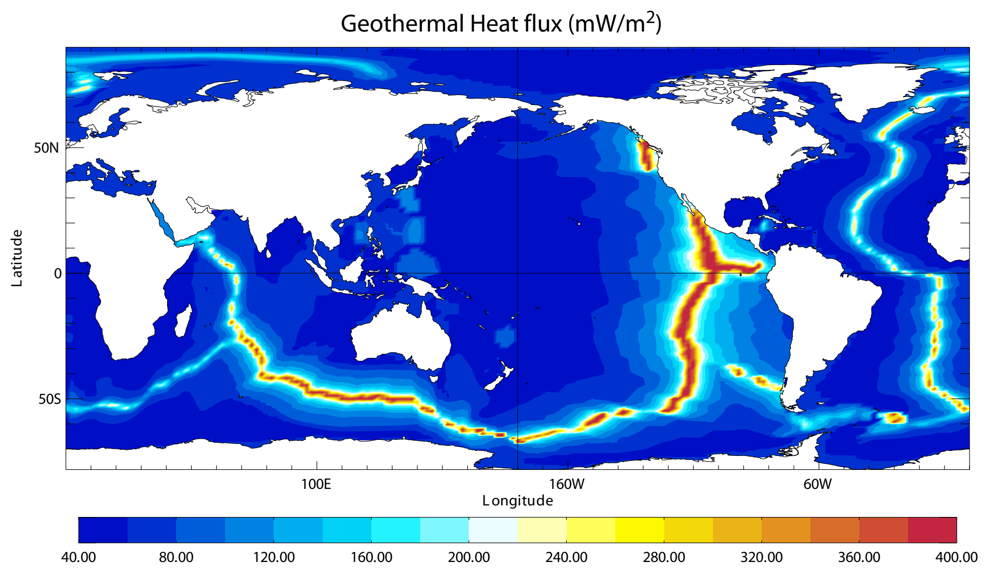


FIG. 4.2 – Geothermal Heat flux (in $mW.m^{-2}$) used by **?**. It is inferred from the age of the sea floor and the formulae of **?**.

4.4.3 Bottom Boundary Condition (*trabbc.F90* - key_bbc)

```
!-----------------------------------------------------------------------
&nambbc        !   bottom temperature boundary condition
!-----------------------------------------------------------------------
   ngeo_flux   =    2      !  geothermal heat flux = 0 no flux considered
                           !                       = 1 constant flux
                           !                       = 2 variable flux (read in geothermal_heating.nc in mW/m2)
   ngeo_flux_const = 86.4e-3  !  Constant value of geothermal heat flux [W/m2]
/
```

Usually it is assumed that there is no exchange of heat or salt through the ocean bottom, $i.e.$ a no flux boundary condition is applied on active tracers at the bottom. This is the default option in *NEMO* , and it is implemented using the masking technique. However, there is a non-zero heat flux across the seafloor that is associated with solid earth cooling. This flux is weak compared to surface fluxes (a mean global value of $\sim 0.1\ W/m^2$ [**?**]), but it is systematically positive and acts on the densest water masses. Taking this flux into account in a global ocean model increases the deepest overturning cell ($i.e.$ the one associated with the Antarctic Bottom Water) by a few Sverdrups [**?**].

The presence or not of geothermal heating is controlled by the namelist parameter *ngeo_flux*. If this parameter is set to 1, a constant geothermal heating is introduced whose value is given by the *ngeo_flux_const*, which is also a namelist parameter. If it is set to 2, a spatially varying geothermal heat flux is introduced which is provided in the geothermal_heating.nc NetCDF file (Fig.4.4.3).

4.5 Bottom Boundary Layer (*trabbl.F90, trabbl_adv.F90*)

```
!-----------------------------------------------------------------------
&nambbl       !   bottom boundary layer scheme
!-----------------------------------------------------------------------
!                         !  diffusive bbl                            ("key_trabbl")
!                         !  advective bbl                            ("key_trabbl_adv")
   atrbbl      =  10000.   !  lateral mixing coefficient in the bbl  [m2/s]
/
```

In a z-coordinate configuration, the bottom topography is represented by a series of discrete steps. This is not adequate to represent gravity driven downslope flows. Such flows arise downstream of sills such as the Strait of Gibraltar, Bab El Mandeb, or Denmark Strait, where dense water formed in marginal seas flows into a basin filled with less dense water. The amount of entrainment that occurs in these gravity plumes is critical in determining the density and volume flux of the densest waters of the ocean, such as Antarctic Bottom Water, or North Atlantic Deep Water. z-coordinate models tend to overestimate the entrainment, because the gravity flow is mixed down vertically by convection as it goes “downstairs” following the step topography, sometimes over a thickness much larger than the thickness of the observed gravity plume. A similar problem occurs in the s-coordinate when the thickness of the bottom level varies in large proportions downstream of a sill [**?**], and the thickness of the plume is not resolved.

The idea of the bottom boundary layer (BBL) parameterisation first introduced by **?** is to allow a direct communication between two adjacent bottom cells at different levels, whenever the densest water is located above the less dense water. The communication can be by a diffusive (diffusive BBL), advective fluxes (advective BBL), or both. In the current implementation of the BBL, only the tracers are modified, not the velocities. Furthermore, it only connects ocean bottom cells, and therefore does not include the improvment proposed by **?**.

4.5.1 Diffusive Bottom Boundary layer (key_bbl_diff)

When applying sigma-diffusion (**key_trabbl** is defined), the diffusive flux between two adjacent cells living at the ocean bottom is given by

$$\mathbf{F}_\sigma = A_l^\sigma \; \nabla_\sigma T \tag{4.18}$$

with ∇_σ the lateral gradient operator taken between bottom cells, and A_l^σ the lateral diffusivity in the BBL. Following **?**, the latter is prescribed with a spatial dependence, $e.g.$ in the conditional form

$$A_l^\sigma(i,j,t) = \begin{cases} A_{bbl} & \text{if} \quad \nabla_\sigma \rho \cdot \nabla H < 0 \\ \\ 0 & \text{otherwise} \end{cases} \tag{4.19}$$

where A_{bbl} is the BBL diffusivity coefficient, given by the namelist parameter *atrbbl*. A_{bbl} is usually set to a value much larger than the one used on lateral mixing in open ocean. Note that in practice, (4.19) constraint is applied separately in the two horizontal directions, and the density gradient in (4.19) is evaluated at $\overline{H}^i$ ($\overline{H}^j$) using the along bottom mean temperature and salinity.

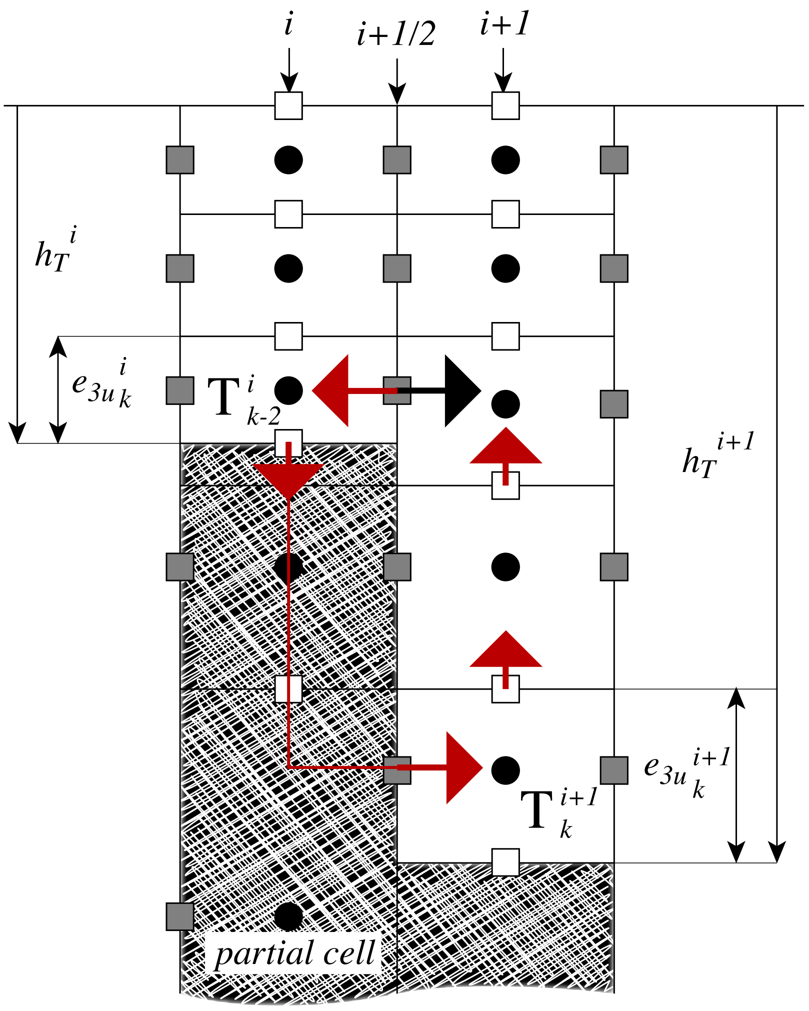


FIG. 4.3 – Advective Bottom Boundary Layer.

4.5.2 Advective Bottom Boundary Layer (key_bbl_adv)

The advective BBL is in fact not only an advective one but include a diffusive component as we chose an upstream scheme to perform the advection within the BBL. The associated diffusion only act in the stream direction and is proportional to the velocity.

When applying sigma-advection (**key_trabbl_adv** defined), the advective flux between two adjacent cells living at the ocean bottom is given by

$$\mathbf{F}_\sigma = \mathbf{U}_h^\sigma \, \overline{T}^\sigma \tag{4.20}$$

with ∇_σ the lateral gradient operator taken between bottom cells, and A_l^σ the lateral diffusivity in the BBL. Following **?**, the latter is prescribed with a spatial dependence, $e.g.$ in the conditional form

$$A_l^\sigma(i,j,t) = \begin{cases} A_{bbl} & \text{if} \quad \nabla_\sigma \rho \cdot \nabla H < 0 \qquad \text{and} \quad \mathbf{U}_h \cdot \nabla H < 0 \\ \\ 0 & \text{otherwise} \end{cases} \tag{4.21}$$

4.6 Tracer damping (*tradmp.F90*)

```
!-----------------------------------------------------------------------
&namtdp        !   tracer newtonian damping                             ('key_tradmp')
!-----------------------------------------------------------------------
   ndmp       =   -1     !  type of damping in temperature and salinity
                         !    ='latitude', damping poleward of 'ndmp' degrees and function
                         !                 of the distance-to-coast. Red and Med Seas as ndmp=-1
                         !    =-1 damping only in Med and Red Seas
   ndmpf      =    1     !  create a damping.coeff NetCDF file (=1) or not (=0)
   nmldmp     =    1     !  type of damping: =0 damping throughout the water column
                         !                   =1 no damping in the mixed layer defined by avt >5cm2/s )
                         !                   =2 no damping in the mixed layer defined rho<rho(surf)+.01 )
   sdmp       =   50.    !  surface time scale for internal damping (days)
   bdmp       =  360.    !  bottom  time scale for internal damping (days)
   hdmp       =  800.    !  depth of transition between sdmp and bdmp (meters)
/
```

In some applications it can be useful to add a Newtonian damping term into the temperature and salinity equations :

$$\begin{aligned} \frac{\partial T}{\partial t} &= \cdots - \gamma \, (T - T_o) \\ \frac{\partial S}{\partial t} &= \cdots - \gamma \, (S - S_o) \end{aligned} \tag{4.22}$$

where γ is the inverse of a time scale, and T_o and S_o are given temperature and salinity fields (usually a climatology). The restoring term is added when **key_tradmp** is defined. It also requires that both **key_temdta** and **key_saldta** are defined ($i.e.$ that T_o and S_o are read). The restoring coefficient S_o is a three-dimensional array initialized by the user in routine *dtacof* also located in module *tradmp.F90*.

The two main cases in which (4.22) is used are *(a)* the specification of the boundary conditions along artificial walls of a limited domain basin and *(b)* the computation of the velocity field associated with a given T-S field (for example to build the initial state of a prognostic simulation, or to use the resulting velocity field for a passive tracer study). The first case applies to regional models that have artificial walls instead of open boundaries. In the vicinity of these walls, S_o takes large values (equivalent to a time scale of a few days) whereas it is zero in the interior of the model domain. The second case corresponds to the use of the robust diagnostic method [**?**]. It allows us to find the velocity field consistent with the model dynamics whilst having a T-S field close to a given climatological field ($T_o - S_o$). The time scale associated with S_o is generally not a constant but spatially varying in order to respect other properties. For example, it is usually set to zero in the mixed layer (defined either on a density or S_o criterion) [**?**] and in the equatorial region [**???**] since these two regions have a short time scale of adjustment ; while smaller

S_o are used in the deep ocean where the typical time scale is long [**?**]. In addition the time scale is reduced (even to zero) along the western boundary to allow the model to reconstruct its own western boundary structure in equilibrium with its physics. The choice of a Newtonian damping acting in the mixed layer or not is controlled by namelist parameter *nmldmp*.

The robust diagnostic method is very efficient in preventing temperature drift in intermediate waters but it produces artificial sources of heat and salt within the ocean. It also has undesirable effects on the ocean convection. It tends to prevent deep convection and subsequent deep-water formation, by stabilising the water column too much.

An example of the computation of S_o for robust diagnostic experiments with the ORCA2 model is provided in the *tradmp.F90* module (subroutines *dtacof* and *cofdis* which compute the coefficient and the distance to the bathymetry, respectively). These routines are provided as examples and can be customised by the user.

4.7 Tracer time evolution (*tranxt.F90*)

```
!-----------------------------------------------------------------------
&namdom        !   space and time domain (bathymetry, mesh, timestep)
!-----------------------------------------------------------------------
   ntopo       =    1      !  compute (=0) or read(=1) the bathymetry file
   e3zps_min   =    5.     !  the thickness of the partial step is set larger than the minimum
   e3zps_rat   =    0.1    !  of e3zps_min and e3zps_rat * e3t   (N.B. 0<e3zps_rat<1)
   nmsh        =    0      !  create (=1) a mesh file (coordinates, scale factors, masks) or not (=0)
   nacc        =    0      !  =1 acceleration of convergence method used, rdt < rdttra(k)
                           !  =0, no acceleration, rdt = rdttra
   atfp        =    0.1    !  asselin time filter parameter
   rdt         = 5760.     !  time step for the dynamics (and tracer if nacc=0)
   rdtmin      = 5760.     !  minimum time step on tracers (used if nacc=1)
   rdtmax      = 5760.     !  maximum time step on tracers (used if nacc=1)
   rdth        =  800.     !  depth variation of tracer time step  (used if nacc=1)
   rdtbt       =   90.     !  barotropic time step (for the split explicit algorithm) ("key_dynspg_ts")
   nclosea     =    0      !  = 0 no closed sea in the model domain
                           !  = 1 closed sea (Black Sea, Caspian Sea, Great US Lakes...)
/
```

The general framework for tracer time stepping is a leap-frog scheme, $i.e.$ a three level centred time scheme associated with a Asselin time filter (cf. §3.4) :

$$\begin{aligned} T^{t+\Delta t} &= T^{t-\Delta t} + 2\,\Delta t\ \mathrm{RHS}_T^t \\ T_f^t &= T^t \quad + \gamma\left[T_f^{t-\Delta t} - 2T^t + T^{t+\Delta t}\right] \end{aligned} \tag{4.23}$$

where RHS_T is the right hand side of the temperature equation, the subscript f denotes filtered values and γ is the Asselin coefficient. γ is initialized as *atfp* (**namelist** parameter). Its default value is *atfp=0.1*.

When the vertical mixing is solved implicitly, the update of the *next* tracer fields is done in module *trazdf.F90*. In this case only the swapping of arrays and the Asselin filtering is done in the *tranxt.F90* module.

In order to prepare for the computation of the *next* time step, a swap of tracer arrays is performed : $T^{t-\Delta t} = T^t$ and $T^t = T_f$.

4.8 Equation of State (*eosbn2.F90*)

```
!-----------------------------------------------------------------------
&nameos        !   ocean physical parameters
!-----------------------------------------------------------------------
   neos        =    0      !  type of equation of state and Brunt-Vaisala frequency
                           !     = 0, UNESCO (formulation of Jackett and McDougall (1994) and of McDougall (1987) )
                           !     = 1, linear: rho(T)   = rau0 * ( 1.028 - ralpha * T )
                           !     = 2, linear: rho(T,S) = rau0 * ( rbeta * S - ralpha * T )
   ralpha      =    2.e-4  !  thermal expension coefficient (neos= 1 or 2)
   rbeta       =    0.001  !  saline  expension coefficient (neos= 2)
/
```

4.8.1 Equation of State (*neos* = 0, 1 or 2)

It is necessary to know the equation of state for the ocean very accurately to determine stability properties (especially the Brunt-Vaisälä frequency), particularly in the deep ocean. The ocean density is a non linear empirical function of *in situ* temperature, salinity and pressure. The reference equation of state is that defined by the Joint Panel on Oceanographic Tables and Standards [**?**]. It was the standard equation of state used in early releases of OPA. However, even though this computation is fully vectorised, it is quite time consuming (15 to 20% of the total CPU time) since it requires the prior computation of the *in situ* temperature from the model *potential* temperature using the [**?**] polynomial for adiabatic lapse rate and a 4^th order Runge-Kutta integration scheme. Since OPA6, we have used the **?** equation of state for seawater instead. It allows the computation of the *in situ* ocean density directly as a function of *potential* temperature relative to the surface (an *NEMO* variable), the practical salinity (another *NEMO* variable) and the pressure (assuming no pressure variation along geopotential surfaces, i.e. the pressure in decibars is approximated by the depth in meters). Both the **?** and **?** equations of state have exactly the same except that the values of the various coefficients have been adjusted by **?** in order to directly use the *potential* temperature instead of the *in situ* one. This reduces the CPU time of the in situ density computation to about 3% of the total CPU time, while maintaining a quite accurate equation of state.

In the computer code, a *true* density d is computed, $i.e.$ the ratio of seawater volumic mass to ρ_o, a reference volumic mass (*rau0* defined in *phycst.F90*, usually $rau0 = 1,020\ Kg/m^3$). The default option (namelist prameter *neos*=0) is the **?** equation of state. Its use is highly recommended. However, for process studies, it is often convenient to use a linear approximation of the density* [2]. Two linear formulations are available : a function of T only (*neos*=1) and a function of both T and S (*neos*=2) :

$$\begin{aligned} d(T) = \rho(T)/\rho_0 &= 1.028 - \alpha\, T \\ d(T,S) = \rho(T,S) &= \quad \beta\, S - \alpha\, T \end{aligned} \tag{4.24}$$

where α and β are the thermal and haline expansion coefficients, and ρ_o, the reference volumic mass, $rau0$. (α and β can be modified through the *ralpha* and *rbeta* namelist parameters). Note that when d is a function of T only (*neos*=1), the salinity is a passive tracer and can be used as such.

[2]* With the linear equation of state there is no longer a distinction between *in situ* and *potential* density. Cabling and thermobaric effects are also removed.

4.8.2 Brunt-Vaisälä Frequency (*neos* = 0, 1 or 2)

An accurate computation of the ocean stability (i.e. of N, the brunt-Vaisälä frequency) is of paramount importance as it is used in several ocean parameterisations (namely TKE, KPP, Richardson number dependent vertical diffusion, enhanced vertical diffusion, non-penetrative convection, iso-neutral diffusion). In particular, one must be aware that N^2 has to be computed with an *in situ* reference. The expression for N^2 depends on the type of equation of state used (*neos* namelist parameter).

For *neos*=0 (**?** equation of state), the **?** polynomial expression is used (with the pressure in decibar approximated by the depth in meters) :

$$N^2 = \frac{g}{e_{3w}}\ \beta\ \left(\alpha/\beta\ \delta_{k+1/2}[T] - \delta_{k+1/2}[S]\right) \tag{4.25}$$

where α (β) is the thermal (haline) expansion coefficient. They are a function of $\overline{T}^{\,k+1/2}$, $\widetilde{S} = \overline{S}^{\,k+1/2} - 35.$, and z_w, with T the *potential* temperature and $\widetilde{S}$ a salinity anomaly. Note that both α and β depend on *potential* temperature and salinity which are averaged at w-points prior to the computation instead of being computed at T-points and then averaged to w-points.

When a linear equation of state is used (*neos*=1 or 2, (4.25) reduces to :

$$N^2 = \frac{g}{e_{3w}} \left(\beta\ \delta_{k+1/2}[S] - \alpha\ \delta_{k+1/2}[T]\right) \tag{4.26}$$

where α and β are the constant coefficients used to defined the linear equation of state (4.24).

4.8.3 Specific Heat (*phycst.F90*)

The specific heat of sea water, C_p, is a function of temperature, salinity and pressure [**?**]. It is only used in the model to convert surface heat fluxes into surface temperature increase and so the pressure dependence is neglected. The dependence on T and S is weak. For example, with $S = 35\,psu$, C_p increases from 3989 to 4002 when T varies from -2 ˚C to 31 ˚C. Therefore, C_p has been chosen as a constant : $C_p = 4.10^3\ J\,Kg^{-1}\,{}^{\circ}K^{-1}$. Its value is set in *phycst.F90* module.

4.8.4 Freezing Point of Seawater (*ocfzpt.F90*)

The freezing point of seawater is a function of salinity and pressure [**?**] :

$$\begin{aligned} T_f(S,p) = &\left(-0.0575 + 1.710523\ 10^{-3}\,\sqrt{S} - 2.154996\ 10^{-4}\,S\right)\ S \\ &-7.53\,10^{-3}\ \ p \end{aligned} \tag{4.27}$$

(4.27) is only used to compute the potential freezing point of sea water ($i.e.$ referenced to the surface $p = 0$), thus the pressure dependent terms in (4.27) (last term) have been dropped. The *before* and *now* surface freezing point is introduced in the code as $fzptb$ and $fzptn$ 2D arrays together with a *now* mask (*freezn*) which takes the value 0 or 1 depending on whether the ocean temperature is above or at the freezing point. Caution : do not confuse *freezn* with the fraction of lead (*frld*) defined in LIM.

4.9 Horizontal Derivative in *zps*-coordinate (*zpshde.F90*)

With partial bottom cells (*ln_zps*=.true.), in general, tracers in horizontally adjacent cells live at different depths. Horizontal gradients of tracers are needed for horizontal diffusion (*traldf.F90* module) and for the hydrostatic pressure gradient (*dynhpg.F90* module) to be active. Before taking horizontal gradients between the tracers next to the bottom, a linear interpolation in the vertical is used to approximate the deeper tracer as if it actually lived at the depth of the shallower tracer point (Fig. 4.9). For example, for temperature in the i-direction the needed interpolated temperature, $\widetilde{T}$, is :

$$\widetilde{T} = \begin{cases} T^{\,i+1} - \dfrac{\left(e_{3w}^{i+1} - e_{3w}^{i}\right)}{e_{3w}^{i+1}}\;\delta_k T^{i+1} & \text{if } e_{3w}^{i+1} \geq e_{3w}^{i} \\[2em] T^{\,i} \quad + \dfrac{\left(e_{3w}^{i+1} - e_{3w}^{i}\right)}{e_{3w}^{i}}\;\delta_k T^{i+1} & \text{if } e_{3w}^{i+1} < e_{3w}^{i} \end{cases}$$

and the resulting forms for the horizontal difference and the horizontal average value of T at a U-point are :

$$\begin{aligned} \delta_{i+1/2} T &= \begin{cases} \widetilde{T} \quad - T^{i} & \text{if } e_{3w}^{i+1} \geq e_{3w}^{i} \\[1.5em] T^{\,i+1} - \widetilde{T} & \text{if } e_{3w}^{i+1} < e_{3w}^{i} \end{cases} \\[1em] \overline{T}^{\,i+1/2} &= \begin{cases} (\widetilde{T} \quad - T^{\,i})/2 & \text{if } e_{3w}^{i+1} \geq e_{3w}^{i} \\[1.5em] (T^{\,i+1} - \widetilde{T})/2 & \text{if } e_{3w}^{i+1} < e_{3w}^{i} \end{cases} \end{aligned} \tag{4.28}$$

The computation of horizontal derivative of tracers as well as of density is performed once for all at each time step in *zpshde.F90* module and stored in shared arrays to be used when needed. It has to be emphasized that the procedure used to compute the interpolated density, $\widetilde{\rho}$, is not the same as that used for T and S. Instead of forming a linear approximation of density, we compute $\widetilde{\rho}$ from the interpolated values of T and S, and the pressure at a u-point (in the equation of state pressure is approximated by depth, see §4.8.1) :

$$\widetilde{\rho} = \rho(\widetilde{T}, \widetilde{S}, z_u) \quad \text{where} \;\; z_u = \min\left(z_T^{i+1}, z_T^{i}\right) \tag{4.29}$$

This is a much better approximation as the variation of ρ with depth (and thus pressure) is highly non-linear with a true equation of state and thus is badly approximated with

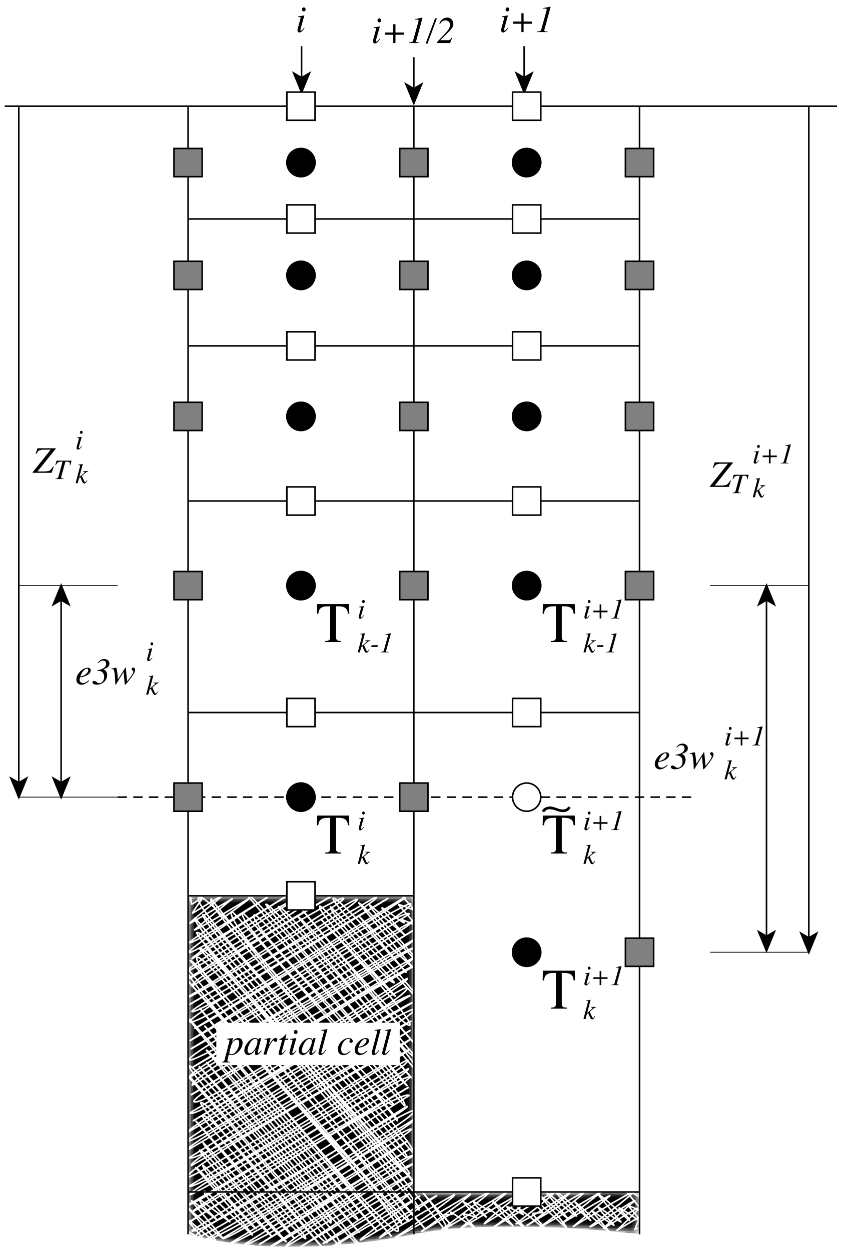


FIG. 4.4 – Discretisation of the horizontal difference and average of tracers in the z-partial step coordinate (*ln_zps*=.true.) in the case $(e3w_k^{i+1} - e3w_k^i) > 0$. A linear interpolation is used to estimate $\widetilde{T}_k^{i+1}$, the tracer value at the depth of the shallower tracer point of the two adjacent bottom T-points. The horizontal difference is then given by : $\delta_{i+1/2} T_k = \widetilde{T}_k^{i+1} - T_k^i$ and the average by : $\overline{T}_k^{i+1/2} = (\widetilde{T}_k^{i+1/2} - T_k^i)/2$.

a linear interpolation. This approximation is used to compute both the horizontal pressure gradient (§5.3) and the slopes of neutral surfaces (§8.2)

Note that in almost all the advection schemes presented in this Chapter, both averaging and differencing operators appear. Yet (4.28) has not been used in these schemes : in contrast to diffusion and pressure gradient computations, no correction for partial steps is applied for advection. The main motivation is to preserve the domain averaged mean variance of the advected field when using the 2^{nd} order centred scheme. Sensitivity of the advection schemes to the way horizontal averages are performed in the vicinity of partial cells should be further investigated in the near future.

Ocean Dynamics (DYN)

Contents

Using the representation described in Chap.3, several semi-discrete space forms of the dynamical equations are available depending on the vertical coordinate used and on the conservation properties of the vorticity term. In all the equations presented here, the masking has been omitted for simplicity. One must be aware that all the quantities are masked fields and that each time a average or difference operator is used, the resulting field is multiplied by a mask.

The prognostic ocean dynamics equation can be summarized as follows :

$$\text{NXT} = \begin{pmatrix} \text{VOR} + \text{KEG} + \text{ZAD} \\ \text{COR} + \text{ADV} \end{pmatrix} + \text{HPG} + \text{SPG} + \text{LDF} + \text{ZDF}$$

NXT stands for next, referring to the time-stepping. The first group of terms on the rhs of the momentum equations corresponds to the Coriolis and advection terms that are decomposed into a vorticity part (VOR), a kinetic energy part (KEG) and, a vertical advection part (ZAD) in the vector invariant formulation or a Coriolis and advection part(COR+ADV) in the flux formulation. The terms following these are the pressure gradient contributions (HPG, Hydrostatic Pressure Gradient, and SPG, Surface Pressure Gradient) ; and contributions from lateral diffusion (LDF) and vertical diffusion (ZDF), which are added to the rhs in the *dynldf.F90* and *dynzdf.F90* modules. The vertical diffusion term includes the surface and bottom stresses. The external forcings and parameterisations require complex inputs (surface wind stress calculation using bulk formulae, estimation of mixing coefficients) that are carried out in modules SBC, LDF and ZDF and are described in Chapters 6, 8 and 9, respectively.

In the present chapter we also describe the diagnostic equations used to compute the horizontal divergence and curl of the velocities (*divcur* module) as well as the vertical velocity (*wzvmod* module).

The different options available to the user are managed by namelist variables. For equation term *ttt*, the logical namelist variables are *ln_dynttt_xxx*, where *xxx* is a 3 or 4 letter acronym corresponding to each optional scheme. If a CPP key is used for this term its name is **key_ttt**. The corresponding code can be found in the *dynttt_xxx* module in the DYN directory, and it is usually computed in the *dyn_ttt_xxx* subroutine.

The user has the option of extracting each tendency term of both the rhs of the 3D momentum equation (**key_trddyn** defined) for output, as described in Chap.10. Furthermore, the tendency terms associated to the 2D barotropic vorticity balance (**key_trdvor** defined) can be derived on-line from the 3D terms.

5.1 Coriolis and Advection : vector invariant form

```
!-----------------------------------------------------------------------
&nam_dynadv    !   formulation of the momentum advection
!-----------------------------------------------------------------------
   ln_dynadv_vec = .true.  !  vector form (T) or flux form (F)
   ln_dynadv_cen2= .false. !  flux form - 2nd order centered scheme
   ln_dynadv_ubs = .false. !  flux form - 3rd order UBS      scheme
/
```

The vector invariant form of the momentum equations is the one most often used in applications of *NEMO* ocean model. The flux form option (see next section) has been introduced since version 2. Coriolis and momentum advection terms are evaluated using a leapfrog scheme, $i.e.$ the velocity appearing in these expressions is centred in time (*now* velocity). At the lateral boundaries either free slip, no slip or partial slip boundary conditions are applied following Chap.7.

5.1.1 Vorticity term (*dynvor.F90*)

```
!-----------------------------------------------------------------------
&nam_dynvor    !   option of physics/algorithm (not control by CPP keys)
!-----------------------------------------------------------------------
   ln_dynvor_ene = .false. !  enstrophy conserving scheme
   ln_dynvor_ens = .true.  !  energy conserving scheme
   ln_dynvor_mix = .false. !  mixed scheme
   ln_dynvor_een = .false. !  energy & enstrophy scheme
/
```

Different discretisations of the vorticity term (*ln_dynvor_xxx*=.true.) are available : conserving potential enstrophy of horizontally non-divergent flow ; conserving horizontal kinetic energy ; or conserving potential enstrophy for the relative vorticity term and horizontal kinetic energy for the planetary vorticity term (see Appendix C). The vorticity terms are given below for the general case, but note that in the full step z-coordinate (**key_zco** is defined), $e_{3u} = e_{3v} = e_{3f}$ so that the vertical scale factors disappear. They are all computed in dedicated routines that can be found in the *dynvor.F90* module.

Enstrophy conserving scheme (*ln_dynvor_ens*=.true.)

In the enstrophy conserving case (ENS scheme), the discrete formulation of the vorticity term provides a global conservation of the enstrophy ($[(\zeta+f)/e_{3f}]^2$ in s-coordinates) for a horizontally non-divergent flow ($i.e.$ $\chi = 0$), but does not conserve the total kinetic energy. It is given by :

$$\left\{ \begin{aligned} &+\frac{1}{e_{1u}} \overline{\left(\frac{\zeta+f}{e_{3f}} \right)}^{\,i} & \overline{\overline{\left(e_{1v} e_{3v} v \right)}}^{\,i, j+1/2} \\ &-\frac{1}{e_{2v}} \overline{\left(\frac{\zeta+f}{e_{3f}} \right)}^{\,j} & \overline{\overline{\left(e_{2u} e_{3u} u \right)}}^{\,i+1/2, j} \end{aligned} \right. \tag{5.1}$$

Energy conserving scheme (*ln_dynvor_ene*=.true.)

The kinetic energy conserving scheme (ENE scheme) conserves the global kinetic energy but not the global enstrophy. It is given by :

$$\left\{ \begin{aligned} &+\frac{1}{e_{1u}} \overline{\left(\frac{\zeta+f}{e_{3f}}\right) \overline{(e_{1v} e_{3v} v)}^{\,i+1/2}}^{\,j} \\ &-\frac{1}{e_{2v}} \overline{\left(\frac{\zeta+f}{e_{3f}}\right) \overline{(e_{2u} e_{3u} u)}^{\,j+1/2}}^{\,i} \end{aligned} \right. \tag{5.2}$$

Mixed energy/enstrophy conserving scheme (*ln_dynvor_mix*=.true.)

The mixed energy/enstrophy conserving scheme (MIX scheme), a mixture of the two previous schemes is used. It consists of the ENS scheme (5.1) to the relative vorticity term, and of the ENE scheme (5.2) applied to the planetary vorticity term.

$$\left\{ \begin{aligned} &+\frac{1}{e_{1u}} \overline{\left(\frac{\zeta}{e_{3f}}\right)}^{\,i} \overline{\overline{(e_{1v}\ e_{3v}\ v)}}^{\,i,j+1/2} - \frac{1}{e_{1u}} \overline{\left(\frac{f}{e_{3f}}\right) \overline{(e_{1v}\ e_{3v}\ v)}^{\,i+1/2}}^{\,j} \\ &-\frac{1}{e_{2v}} \overline{\left(\frac{\zeta}{e_{3f}}\right)}^{\,j} \overline{\overline{(e_{2u}\ e_{3u}\ u)}}^{\,i+1/2,j} + \frac{1}{e_{2v}} \overline{\left(\frac{f}{e_{3f}}\right) \overline{(e_{2u}\ e_{3u}\ u)}^{\,j+1/2}}^{\,i} \end{aligned} \right. \tag{5.3}$$

Energy and enstrophy conserving scheme (*ln_dynvor_een*=.true.)

In the energy and enstrophy conserving scheme (EEN scheme), the vorticity term is evaluated using the vorticity advection scheme of **?**. This scheme conserves both total energy and potential enstrophy in the limit of horizontally nondivergent flow ($i.e.\ \chi = 0$). While EEN is more complicated than ENS or ENE and does not conserve potential enstrophy and total energy in general flow, it tolerates arbitrarily thin layers. This feature is essential for z-coordinate with partial step.

The **?** vorticity advection scheme for a single layer is modified for spherical coordinates as described by **?** to obtain the EEN scheme. The potential vorticity, defined at an f-point, is :

$$q_f = \frac{\zeta+f}{e_{3f}} \tag{5.4}$$

where the relative vorticity is defined by (5.29), the Coriolis parameter is given by $f = 2\,\Omega\ \sin\varphi_f$ and the layer thickness at f-points is :

$$e_{3f} = \overline{\overline{e_{3t}}}^{\,i+1/2,j+1/2} \tag{5.5}$$

Note that a key point in (5.5) is that the averaging in **i-** and **j-** directions uses the masked vertical scale factor but is always divided by 4, not by the sum of the mask at T-point. This preserves the continuity of e_{3f} when one or more of the neighbouring e_{3T} tends to zero and extends by continuity the value of e_{3f} in the land areas.

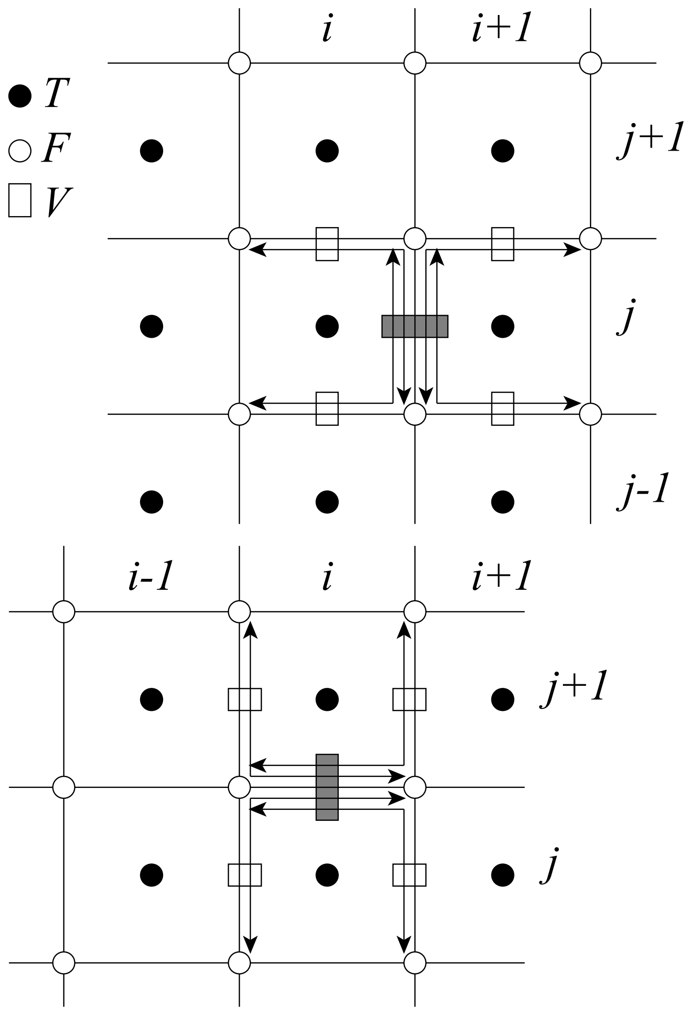


FIG. 5.1 – Triads used in the energy and enstrophy conserving scheme (een) for u-component (upper panel) and v-component (lower panel).

The vorticity terms are represented as :

$$
\left\{
\begin{aligned}
+q\, e_3\, v \equiv +\frac{1}{e_{1u}} & \left[\begin{aligned} & a^i_{j+1/2} \left(e_{1v} e_{3v}\ v\right)^{i+1/2}_{j+1} + b^i_{j+1/2} \left(e_{1v} e_{3v}\ v\right)^{i-1/2}_{j+1} \\ & + c^i_{j-1/2} \left(e_{1v} e_{3v}\ v\right)^{i+1/2}_{j} + d^i_{j+1/2} \left(e_{1v} e_{3v}\ v\right)^{i+1/2}_{j+1} \end{aligned} \right] \\
-q\, e_3\, u \equiv -\frac{1}{e_{2v}} & \left[\begin{aligned} & a^i_{j-1/2} \left(e_{2u} e_{3u}\ u\right)^{i+1/2}_{j+1} + b^{i+1}_{j-1/2} \left(e_{2u} e_{3u}\ u\right)^{i+1}_{j+1/2} \\ & + c^{i+1}_{j+1/2} \left(e_{2u} e_{3u}\ u\right)^{i+1}_{j+1/2} + d^i_{j+1/2} \left(e_{2u} e_{3u}\ u\right)^{i}_{j+1/2} \end{aligned} \right]
\end{aligned}
\right.
\tag{5.6}
$$

where a, b, c and d are the following triad combinations of the neighbouring potential vorticities (Fig. 5.1.1) :

$$\left\{\begin{aligned} {a^i}_{j+1/2} &= \frac{1}{12}\left(q^{i+1}_{j+1/2} + q^{i}_{j+1/2} + q^{i}_{j-1/2}\right) \\ {b^i}_{j+1/2} &= \frac{1}{12}\left(q^{i-1}_{j+1/2} + q^{i}_{j+1/2} + q^{i}_{j-1/2}\right) \\ {c^i}_{j+1/2} &= \frac{1}{12}\left(q^{i-1}_{j-1/2} + q^{i}_{j+1/2} + q^{i}_{j-1/2}\right) \\ {d^i}_{j+1/2} &= \frac{1}{12}\left(q^{i+1}_{j-1/2} + q^{i}_{j+1/2} + q^{i}_{j-1/2}\right) \end{aligned}\right. \tag{5.7}$$

5.1.2 Kinetic Energy Gradient term (*dynkeg.F90*)

As demonstarted in Appendix C, there is a single discrete formulation of the kinetic energy gradient term that, together with the formulation chosen for the vertical advection (see below), conserves the total kinetic energy :

$$\left\{\begin{aligned} &-\frac{1}{2\;e_{1u}}\;\delta_{i+1/2}\left[\overline{u^2}^{\,i} + \overline{v^2}^{\,j}\right] \\ &-\frac{1}{2\;e_{2v}}\;\delta_{j+1/2}\left[\overline{u^2}^{\,i} + \overline{v^2}^{\,j}\right] \end{aligned}\right. \tag{5.8}$$

5.1.3 Vertical advection term (*dynzad.F90*)

The discrete formulation of the vertical advection, together with the formulation chosen for the gradient of kinetic energy (KE) term, conserves the total kinetic energy. Indeed, the change of KE due to the vertical advection is exactly balanced by the change of KE due to the gradient of KE (see Appendix C).

$$\left\{\begin{aligned} &-\frac{1}{e_{1u}\,e_{2u}\,e_{3u}}\;\overline{\overline{e_{1T}\,e_{2T}\,w}^{\,i+1/2}\;\delta_{k+1/2}\left[u\right]}^{\,k} \\ &-\frac{1}{e_{1v}\,e_{2v}\,e_{3v}}\;\overline{\overline{e_{1T}\,e_{2T}\,w}^{\,j+1/2}\;\delta_{k+1/2}\left[u\right]}^{\,k} \end{aligned}\right. \tag{5.9}$$

5.2 Coriolis and Advection : flux form

```
!-----------------------------------------------------------------------
&nam_dynadv    !   formulation of the momentum advection
!-----------------------------------------------------------------------
   ln_dynadv_vec = .true.  !  vector form (T) or flux form (F)
   ln_dynadv_cen2= .false. !  flux form - 2nd order centered scheme
   ln_dynadv_ubs = .false. !  flux form - 3rd order UBS      scheme
/
```

In the flux form (as in the vector invariant form), the Coriolis and momentum advection terms are evaluated using a leapfrog scheme, $i.e.$ the velocity appearing in their expressions is centred in time (*now* velocity). At the lateral boundaries either free slip, no slip or partial slip boundary conditions are applied following Chap.7.

5.2.1 Coriolis plus curvature metric terms (*dynvor.F90*)

In flux form, the vorticity term reduces to a Coriolis term in which the Coriolis parameter has been modified to account for the ”metric” term. This altered Coriolis parameter is thus discretised at f-points. It is given by :

$$\begin{aligned} f+\frac{1}{e_1 e_2}\left(v\frac{\partial e_2}{\partial i}-u\frac{\partial e_1}{\partial j}\right) \\ \equiv f+\frac{1}{e_{1f}\,e_{2f}}\left(\overline{v}^{\,i+1/2}\delta_{i+1/2}\left[e_{2u}\right]-\overline{u}^{\,j+1/2}\delta_{j+1/2}\left[e_{1u}\right]\right) \end{aligned} \tag{5.10}$$

Any of the (5.1), (5.2) and (5.6) schemes can be used to compute the product of the Coriolis parameter and the vorticity. However, the energy-conserving scheme (5.6) has exclusively been used to date. This term is evaluated using a leapfrog scheme, $i.e.$ the velocity is centred in time (*now* velocity).

5.2.2 Flux form Advection term (*dynadv.F90*)

The discrete expression of the advection term is given by :

$$\left\{\begin{aligned} &\frac{1}{e_{1u}\,e_{2u}\,e_{3u}}\left(\delta_{i+1/2}\left[\overline{e_{2u}\,e_{3u}\,u}^{\,i}\;u_T\right]+\delta_j\left[\overline{e_{1u}\,e_{3u}\,v}^{\,i+1/2}\;u_F\right]\right. \\ &\qquad\qquad\left.+\delta_k\left[\overline{e_{1w}\,e_{2w}w}^{\,i+1/2}\;u_{uw}\right]\right) \\ &\frac{1}{e_{1v}\,e_{2v}\,e_{3v}}\left(\delta_i\left[\overline{e_{2u}\,e_{3u}\,u}^{\,j+1/2}\;v_F\right]+\delta_{j+1/2}\left[\overline{e_{1u}\,e_{3u}\,v}^{\,i}\;v_T\right]\right. \\ &\qquad\qquad\left.+\delta_k\left[\overline{e_{1w}\,e_{2w}\,w}^{\,j+1/2}\;v_{vw}\right]\right) \end{aligned}\right. \tag{5.11}$$

Two advection schemes are available : a 2^{nd} order centered finite difference scheme, CEN2, or a 3^{rd} order upstream biased scheme, UBS. The latter is described in **?**. The schemes are selected using the namelist logicals *ln_dynadv_cen2* and *ln_dynadv_ubs*. In flux form, the schemes differ by the choice of a space and time interpolation to define the value of u and v at the centre of each face of u- and v-cells, $i.e.$ at the T-, f-, and uw-points for u and at the f-, T- and vw-points for v.

2^{nd} order centred scheme (cen2) (*ln_dynadv_cen2*=.true.)

In the centered 2^{nd} order formulation, the velocity is evaluated as the mean of the two neighbouring points :

$$\begin{cases} u_T^{cen2} = \overline{u}^i & u_F^{cen2} = \overline{u}^{j+1/2} & u_{uw}^{cen2} = \overline{u}^{k+1/2} \\ v_F^{cen2} = \overline{v}^{i+1/2} & v_F^{cen2} = \overline{v}^j & v_{vw}^{cen2} = \overline{v}^{k+1/2} \end{cases} \tag{5.12}$$

The scheme is non diffusive (i.e. conserves the kinetic energy) but dispersive ($i.e.$ it may create false extrema). It is therefore notoriously noisy and must be used in conjunction with an explicit diffusion operator to produce a sensible solution. The associated time-stepping is performed using a leapfrog scheme in conjunction with an Asselin time-filter, so u and v are the *now* velocities.

Upstream Biased Scheme (UBS) (*ln_dynadv_ubs*=.true.)

The UBS advection scheme is an upstream biased third order scheme based on an upstream-biased parabolic interpolation. For example, the evaluation of u_T^{ubs} is done as follows :

$$u_T^{ubs} = \overline{u}^i - \frac{1}{6} \begin{cases} u"_{i-1/2} & \text{if } \overline{e_{2u}\, e_{3u}\, u}^i \geqslant 0 \\ u"_{i+1/2} & \text{if } \overline{e_{2u}\, e_{3u}\, u}^i < 0 \end{cases} \tag{5.13}$$

where $u"_{i+1/2} = \delta_{i+1/2}\left[\delta_i\left[u\right]\right]$. This results in a dissipatively dominant ($i.e.$ hyper-diffusive) truncation error [**?**]. The overall performance of the advection scheme is similar to that reported in **?**. It is a relatively good compromise between accuracy and smoothness. It is not a *positive* scheme, meaning that false extrema are permitted. But the amplitudes of the false extrema are significantly reduced over those in the centred second order method.

The UBS scheme is not used in all directions. In the vertical, the centred 2^{nd} order evaluation of the advection is preferred, $i.e.$ u_{uw}^{ubs} and u_{vw}^{ubs} in (5.12) are used. UBS is diffusive and is associated with vertical mixing of momentum.

For stability reasons, the first term in (5.13), which corresponds to a second order centred scheme, is evaluated using the *now* velocity (centred in time), while the second term, which is the diffusive part of the scheme, is evaluated using the *before* velocity (forward in time). This is discussed by **?** in the context of the Quick advection scheme.

Note that the UBS and Quadratic Upstream Interpolation for Convective Kinematics (QUICK) schemes only differ by one coefficient. Substituting $1/6$ with $1/8$ in (5.13) leads to the QUICK advection scheme [**?**]. This option is not available through a namelist parameter, since the $1/6$ coefficient is hard coded. Nevertheless it is quite easy to make the substitution in *dynadv_ubs.F90* module and obtain a QUICK scheme.

Note also that in the current version of *dynadv_ubs.F90*, there is also the possibility of using a 4^{th} order evaluation of the advective velocity as in ROMS. This is an error and should be suppressed soon.

5.3 Hydrostatic pressure gradient (*dynhpg.F90*)

```
!-----------------------------------------------------------------------
&nam_dynhpg    !   Hydrostatic pressure gradient option
!-----------------------------------------------------------------------
   ln_hpg_zco  = .false.   !  z-coordinate - full steps
   ln_hpg_zps  = .true.    !  z-coordinate - partial steps (interpolation)
   ln_hpg_sco  = .false.   !  s-coordinate (standard jacobian formulation)
   ln_hpg_hel  = .false.   !  s-coordinate (helsinki modification)
   ln_hpg_wdj  = .false.   !  s-coordinate (weighted density jacobian)
   ln_hpg_djc  = .false.   !  s-coordinate (Density Jacobian with Cubic polynomial)
   ln_hpg_rot  = .false.   !  s-coordinate (ROTated axes scheme)
   gamm        = 0.e0      !  weighting coefficient (wdj scheme)
/
```

```
!-----------------------------------------------------------------------
&namflg        !   algorithm flags (algorithm not control by CPP keys)
!-----------------------------------------------------------------------
   ln_dynhpg_imp = .false. !  hydrostatic pressure gradient: semi-implicit time scheme  (T)
                           !                                 centered      time scheme  (F)
   nn_dynhpg_rst =  0      !  add dynhpg implicit variables in restart ot not (1/0)
/
```

The key distinction between the different algorithms used for the hydrostatic pressure gradient is the vertical coordinate used, since HPG is a *horizontal* pressure gradient, $i.e.$ computed along geopotential surfaces. As a result, any tilt of the surface of the computational levels will require a specific treatment to compute the hydrostatic pressure gradient.

The hydrostatic pressure gradient term is evaluated either using a leapfrog scheme, $i.e.$ the density appearing in its expression is centred in time (*now* rho), or a semi-implcit scheme. At the lateral boundaries either free slip, no slip or partial slip boundary conditions are applied.

5.3.1 z-coordinate with full step (*ln_dynhpg_zco*=.true.)

The hydrostatic pressure can be obtained by integrating the hydrostatic equation vertically from the surface. However, the pressure is large at great depth while its horizontal gradient is several orders of magnitude smaller. This may lead to large truncation errors in the pressure gradient terms. Thus, the two horizontal components of the hydrostatic pressure gradient are computed directly as follows :

for $k = km$ (surface layer, $jk = 1$ in the code)

$$\left\{ \begin{aligned} \delta_{i+1/2} \left[p^h \right] \Big|_{k=km} &= \frac{1}{2} g \;\; \delta_{i+1/2} \left[e_{3w} \; \rho \right] \big|_{k=km} \\ \delta_{j+1/2} \left[p^h \right] \Big|_{k=km} &= \frac{1}{2} g \;\; \delta_{j+1/2} \left[e_{3w} \; \rho \right] \big|_{k=km} \end{aligned} \right. \tag{5.14}$$

for $1 < k < km$ (interior layer)

$$\left\{ \begin{aligned} \delta_{i+1/2} \left[p^h \right] \Big|_{k} &= \delta_{i+1/2} \left[p^h \right] \Big|_{k-1} + \frac{1}{2} \; g \;\; \delta_{i+1/2} \left[e_{3w} \; \overline{\rho}^{k+1/2} \right] \Big|_{k} \\ \delta_{j+1/2} \left[p^h \right] \Big|_{k} &= \delta_{j+1/2} \left[p^h \right] \Big|_{k-1} + \frac{1}{2} \; g \;\; \delta_{j+1/2} \left[e_{3w} \; \overline{\rho}^{k+1/2} \right] \Big|_{k} \end{aligned} \right. \tag{5.15}$$

Note that the $1/2$ factor in (5.14) is adequate because of the definition of e_{3w} as the vertical derivative of the scale factor at the surface level ($z = 0$).

5.3.2 z-coordinate with partial step (*ln_dynhpg_zps*=.true.)

With partial bottom cells, tracers in horizontally adjacent cells generally live at different depths. Before taking horizontal gradients between these tracer points, a linear interpolation is used to approximate the deeper tracer as if it actually lived at the depth of the shallower tracer point.

Apart from this modification, the horizontal hydrostatic pressure gradient evaluated in the z-coordinate with partial step is exactly as in the pure z-coordinate case. As explained in detail in section §4.9, the nonlinearity of pressure effects in the equation of state is such that it is better to interpolate temperature and salinity vertically before computing the density. Horizontal gradients of temperature and salinity are needed for the TRA modules, which is the reason why the horizontal gradients of density at the deepest model level are computed in module *zpsdhe.F90* located in the TRA directory and described in §4.9.

5.3.3 s- and z-s-coordinates

Pressure gradient formulations in s-coordinate have been the subject of a vast literature ($e.g.$, **??**). A number of different pressure gradient options are coded, but they are not yet fully documented or tested.

• Traditional coding (see for example **?** : (*ln_dynhpg_sco*=.true., *ln_dynhpg_hel*=.true.)

$$\left\{ \begin{aligned} &-\frac{1}{\rho_o\, e_{1u}}\;\delta_{i+1/2}\left[p^h\right]+\frac{g\;\overline{\rho}^{\,i+1/2}}{\rho_o\, e_{1u}}\;\delta_{i+1/2}\left[z_T\right] \\ &-\frac{1}{\rho_o\, e_{2v}}\;\delta_{j+1/2}\left[p^h\right]+\frac{g\;\overline{\rho}^{\,j+1/2}}{\rho_o\, e_{2v}}\;\delta_{j+1/2}\left[z_T\right] \end{aligned} \right. \tag{5.16}$$

Where the first term is the pressure gradient along coordinates, computed as in (5.14) - (5.15), and z_T is the depth of the T-point evaluated from the sum of the vertical scale factors at the w-point (e_{3w}). The version *ln_dynhpg_hel*=.true. has been added by Aike Beckmann and involves a redefinition of the relative position of T-points relative to w-points.

• Weighted density Jacobian (WDJ) [**?**] (*ln_dynhpg_wdj*=.true.)

• Density Jacobian with cubic polynomial scheme (DJC) [**?**] (*ln_dynhpg_djc*=.true.)

• Rotated axes scheme (rot) [**?**] (*ln_dynhpg_rot*=.true.)

Note that expression (5.16) is used when the variable volume formulation is activated (**key_vvl**) because in that case, even with a flat bottom, the coordinate surfaces are not horizontal but follow the free surface [**?**]. The other pressure gradient options are not yet available.

5.3.4 Time-scheme (*ln_dynhpg_imp*=.true./.false.)

The default time differencing scheme used for the horizontal pressure gradient is a leapfrog scheme and therefore the density used in all discrete expressions given above is the *now* density, computed from the *now* temperature and salinity. In some specific cases (usually high resolution simulations over an ocean domain which includes weakly stratified regions) the physical phenomenum that controls the time-step is internal gravity waves (IGWs). A semi-implicit scheme for doubling the stability limit associated with IGWs can be used [**??**]. It involves the evaluation of the hydrostatic pressure gradient as an average over the three time levels $t - \Delta t$, t, and $t + \Delta t$ ($i.e.$ *before*, *now* and *after* time-steps), rather than at central time level t only, as in the standard leapfrog scheme.

• leapfrog scheme (*ln_dynhpg_imp*=.true.) :

$$\frac{u^{t+\Delta t} - u^{t-\Delta t}}{2\Delta t} = \; \cdots \; - \frac{1}{\rho_o \, e_{1u}} \delta_{i+1/2} \left[p_h^t \right] \tag{5.17}$$

• semi-implicit scheme (*ln_dynhpg_imp*=.true.) :

$$\frac{u^{t+\Delta t} - u^{t-\Delta t}}{2\Delta t} = \; \cdots \; - \frac{1}{\rho_o \, e_{1u}} \delta_{i+1/2} \left[\frac{p_h^{t+\Delta t} + 2 p_h^t + p_h^{t-\Delta t}}{4} \right] \tag{5.18}$$

The semi-implicit time scheme (5.18) is made possible without significant additional computation since the density can be updated to time level $t + \Delta t$ before computing the horizontal hydrostatic pressure gradient. It can be easily shown that the stability limit associated with the hydrostatic pressure gradient doubles using (5.18) compared to that using the standard leapfrog scheme (5.17). Note that (5.18) is equivalent to applying a time filter to the pressure gradient to eliminate high frequency IGWs. Obviously, when using (5.18), the doubling of the time-step is achievable only if no other factors control the time-step, such as the stability limits associated with advection or diffusion.

In practice, the semi-implicit scheme is used when *ln_dynhpg_imp*=.true.. In this case, we choose to apply the time filter to temperature and salinity used in the equation of state, instead of applying it to the hydrostatic pressure or to the density, so that no additional storage array has to be defined. The density used to compute the hydrostatic pressure gradient (whatever the formulation) is evaluated as follows :

$$\rho^t = \rho(\widetilde{T}, \widetilde{S}, z_T) \quad \text{with} \quad \widetilde{\cdot} = \frac{\cdot^{t+\Delta t} + 2 \; \cdot^{t} + \; \cdot^{t-\Delta t}}{4} \tag{5.19}$$

Note that in the semi-implicit case, it is necessary to save the filtered density, an extra three-dimensional field, in the restart file to restart the model with exact reproducibility. This option is controlled by the namelist parameter *nn_dynhpg_rst*=.true..

5.4 Surface pressure gradient (*dynspg.F90*)

```
!-----------------------------------------------------------------------
!nam_dynspg    !   surface pressure gradient   (CPP key only)
!-----------------------------------------------------------------------
!                        !  Non-linear free surface                 ("key_vvl")
!                        !  explicit free surface                   ("key_dynspg_exp")
!                        !  filtered free surface                   ("key_dynspg_flt")
!                        !  split-explicit free surface             ("key_dynspg_ts")
!                        !  rigid-lid                               ("key_dynspg_rl")
/
```

The form of the surface pressure gradient term is dependent on the representation of the free surface (§2.2). The main distinction is between the fixed volume case (linear free surface or rigid lid) and the variable volume case (nonlinear free surface, **key_vvl** is defined). In the linear free surface case (§2.2.2) and the rigid lid case (§2.2.3), the vertical scale factors e_3 are fixed in time, whilst in the nonlinear case (§2.2.2) they are time-dependent. With both linear and nonlinear free surface, external gravity waves are allowed in the equations, which imposes a very small time step when an explicit time stepping is used. Two methods are proposed to allow a longer time step for the three-dimensional equations : the filtered free surface method, which involves a modification of the continuous equations (see (2.6)), and the split-explicit free surface method described below. The extra term introduced in the filtered method is calculated implicitly, so that the update of the $next$ velocities is done in module *dynspg_flt.F90* and not in *dynnxt.F90*.

5.4.1 Linear free surface formulation (key_exp or _ts or _flt)

In the linear free surface formulation, the sea surface height is assumed to be small compared to the thickness of the ocean levels, so that (a) the time evolution of the sea surface height becomes a linear equation, and (b) the thickness of the ocean levels is assumed to be constant with time. As mentioned in (§2.2.2) the linearization affects the conservation of tracers.

Explicit (key_dynspg_exp)

In the explicit free surface formulation, the model time step is chosen to be small enough to describe the external gravity waves (typically a few tens of seconds). The sea surface height is given by :

$$\frac{\partial \eta}{\partial t} \equiv \frac{\text{EMP}}{\rho_w} + \frac{1}{e_{1T}e_{2T}} \sum_k \left(\delta_i \left[e_{2u} e_{3u} u \right] + \delta_j \left[e_{1v} e_{3v} v \right] \right) \tag{5.20}$$

where EMP is the surface freshwater budget, expressed in Kg/m^2/s (which is equal to mm/s), and ρ_w=1,000 Kg/m^3 is the volumic mass of pure water. If river runoff is expressed as a surface freshwater flux (see §6) then EMP can be written as the evaporation minus precipitation, minus the river runoff. The sea-surface height is evaluated using a leapfrog scheme in combination with an Asselin time filter, $i.e.$ the velocity appearing in (5.20) is centred in time (*now* velocity).

The surface pressure gradient, also evaluated using a leap-frog scheme, is then simply given by :

$$\left\{ \begin{aligned} & -\frac{1}{e_{1u}} \, \delta_{i+1/2} \left[\eta \right] \\ & -\frac{1}{e_{2v}} \, \delta_{j+1/2} \left[\eta \right] \end{aligned} \right. \tag{5.21}$$

Consistent with the linearization, a factor of $\left. \rho \right|_{k=1} / \rho_o$ is omitted in (5.21).

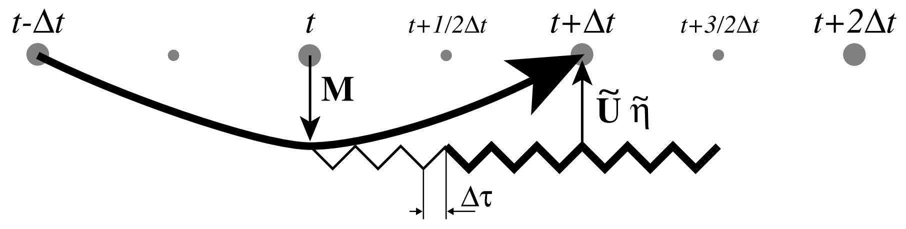


FIG. 5.2 – Schematic of the split-explicit time stepping scheme for the external and internal modes. Time increases to the right. Internal mode time steps (which are also the model time steps) are denoted by $t-\Delta t$, t, $t+\Delta t$, and $t+2\Delta t$. The curved line represents a leap-frog time step, and the smaller time steps $N\Delta t_e = \frac{3}{2}\Delta t$ are denoted by the zig-zag line. The vertically integrated forcing **M**(t) computed at the model time step t represents the interaction between the external and internal motions. While keeping **M** and freshwater forcing field fixed, a leap-frog integration carries the external mode variables (surface height and vertically integrated velocity) from t to $t+\frac{3}{2}\Delta t$ using N external time steps of length Δt_e. Time averaging the external fields over the $\frac{2}{3}N+1$ time steps (endpoints included) centers the vertically integrated velocity and the sea surface height at the model timestep $t+\Delta t$. These averaged values are used to update **M**(t) with both the surface pressure gradient and the Coriolis force, therefore providing the $t+\Delta t$ velocity. The model time stepping scheme can then be achieved by a baroclinic leap-frog time step that carries the surface height from $t-\Delta t$ to $t+\Delta t$.

Split-explicit time-stepping (key_dynspg_ts)

```
!-----------------------------------------------------------------------
&namdom        !   space and time domain (bathymetry, mesh, timestep)
!-----------------------------------------------------------------------
   ntopo       =    1      !  compute (=0) or read(=1) the bathymetry file
   e3zps_min   =    5.     !  the thickness of the partial step is set larger than the minimum
   e3zps_rat   =    0.1    !  of e3zps_min and e3zps_rat * e3t   (N.B. 0<e3zps_rat<1)
   nmsh        =    0      !  create (=1) a mesh file (coordinates, scale factors, masks) or not (=0)
   nacc        =    0      !  =1 acceleration of convergence method used, rdt < rdttra(k)
                           !  =0, no acceleration, rdt = rdttra
   atfp        =    0.1    !  asselin time filter parameter
   rdt         = 5760.     !  time step for the dynamics (and tracer if nacc=0)
   rdtmin      = 5760.     !  minimum time step on tracers (used if nacc=1)
   rdtmax      = 5760.     !  maximum time step on tracers (used if nacc=1)
   rdth        =  800.     !  depth variation of tracer time step  (used if nacc=1)
   rdtbt       =   90.     !  barotropic time step (for the split explicit algorithm) ("key_dynspg_ts")
   nclosea     =    0      !  = 0 no closed sea in the model domain
                           !  = 1 closed sea (Black Sea, Caspian Sea, Great US Lakes...)
/
```

The split-explicit free surface formulation used in *NEMO* follows the one proposed by **?**. The general idea is to solve the free surface equation with a small time step *rdtbt*, while the three dimensional prognostic variables are solved with a longer time step that is a multiple of *rdtbt* (Fig.5.4.1).

The split-explicit formulation has a damping effect on external gravity waves, which is weaker damping than for the filtered free surface but still significant as shown by **?** in the case of an analytical barotropic Kelvin wave.

Filtered formulation (key_dynspg_flt)

The filtered formulation follows the **?** implementation. The extra term introduced in the equations (see §I.2.2) is solved implicitly. The elliptic solvers available in the code are documented in §10. The amplitude of the extra term is given by the namelist variable *rnu*. The default value is 1, as recommended by **?**

5.4.2 Non-linear free surface formulation (key_vvl)

In the non-linear free surface formulation, the variations of volume are fully taken into account. This option is presented in a report [**?**] available on the *NEMO* web site. The three time-stepping methods (explicit, split-explicit and filtered) are the same as in §5.4.1 except that the ocean depth is now time-dependent. In particular, this means that in the filtered case, the matrix to be inverted has to be recomputed at each time-step.

5.4.3 Rigid-lid formulation (key_dynspg_rl)

With the rigid lid formulation, an elliptic equation has to be solved for the barotropic streamfunction. For consistency this equation is obtained by taking the discrete curl of the discrete vertical sum of the discrete momentum equation :

$$\frac{1}{\rho_o}\nabla_h p_s \equiv \left(\begin{array}{c} \overline{M}_u + \frac{1}{H\, e_2}\delta_j \left[\partial_t \psi\right] \\ \\ \overline{M}_v - \frac{1}{H\, e_1}\delta_i \left[\partial_t \psi\right] \end{array} \right) \tag{5.22}$$

Here $\mathbf{M} = (M_u, M_v)$ represents the collected contributions of nonlinear, viscous and hydrostatic pressure gradient terms in (2.1a) and the overbar indicates a vertical average over the whole water column (i.e. from $z = -H$, the ocean bottom, to $z = 0$, the rigid-lid). The time derivative of ψ is the solution of an elliptic equation :

$$\begin{aligned} \delta_{i+1/2}\left[\frac{e_{2v}}{H_v\; e_{1v}}\delta_i\left[\partial_t\psi\right]\right] + \delta_{j+1/2}\left[\frac{e_{1u}}{H_u\; e_{2u}}\delta_j\left[\partial_t\psi\right]\right] \\ = \delta_{i+1/2}\left[e_{2v}M_v\right] - \delta_{j+1/2}\left[e_{1u}M_u\right] \end{aligned} \tag{5.23}$$

The elliptic solvers available in the code are documented in §10). The boundary conditions must be given on all separate landmasses (islands), which is done by integrating the vorticity equation around each island. This requires identifying each island in the bathymetry file, a cumbersome task. This explains why the rigid lid option is not recommended with complex domains such as the global ocean. Parameters jpisl (number of islands) and jpnisl (maximum number of points per island) of the *par_oce.h90* file are related to this option.

5.5 Lateral diffusion term (*dynldf.F90*)

```
!-----------------------------------------------------------------------
&nam_dynldf    !   lateral diffusion on momentum
!-----------------------------------------------------------------------
!                              !  Type of the operator :
   ln_dynldf_lap    =  .true.   !     laplacian operator
   ln_dynldf_bilap  =  .false.  !     bilaplacian operator
!                              !  Direction of action  :
   ln_dynldf_level  =  .false.  !     iso-level
   ln_dynldf_hor    =  .true.   !     horizontal (geopotential)        (require "key_ldfslp" in s-coord.)
   ln_dynldf_iso    =  .false.  !     iso-neutral                      (require "key_ldfslp")
                               !  Coefficient
   rn_ahm_0_lap     = 40000.    !     horizontal laplacian eddy viscosity   [m2/s]
   rn_ahmb_0        =     0.    !     background eddy viscosity for ldf_iso [m2/s]
   rn_ahm_0_blp     =     0.    !     horizontal bilaplacian eddy viscosity [m4/s]
/
```

The options available for lateral diffusion are for the choice of laplacian (rotated or not) or biharmonic operators. The coefficients may be constant or spatially variable ; the description of the coefficients is found in the chapter on lateralphysics (Chap.8). The lateral diffusion of momentum is evaluated using a forward scheme, i.e. the velocity appearing in its expression is the *before* velocity in time, except for the pure vertical component that appears when a tensor of rotation is used. This latter term is solved implicitly together with the vertical diffusion term (see §3.4)

At the lateral boundaries either free slip, no slip or partial slip boundary conditions are applied according to the user's choice (see Chap.7).

5.5.1 Iso-level laplacian operator (*ln_dynldf_lap*=.true.)

For lateral iso-level diffusion, the discrete operator is :

$$\left\{\begin{aligned} D_u^{l\mathbf{U}} &= \frac{1}{e_{1u}}\delta_{i+1/2}\left[A_T^{lm}\;\chi\right] - \frac{1}{e_{2u}\,e_{3u}}\delta_j\left[A_f^{lm}\;e_{3f}\zeta\right] \\ \\ D_v^{l\mathbf{U}} &= \frac{1}{e_{2v}}\delta_{j+1/2}\left[A_T^{lm}\;\chi\right] + \frac{1}{e_{1v}\,e_{3v}}\delta_i\left[A_f^{lm}\;e_{3f}\zeta\right] \end{aligned}\right. \tag{5.24}$$

As explained in §2.6.2, this formulation (as the gradient of a divergence and curl of the vorticity) preserves symmetry and ensures a complete separation between the vorticity and divergence parts. Note that in the full step z-coordinate (**key_zco** is defined), $e_{3u} = e_{3v} = e_{3f}$ so that they cancel in the rotational part of (5.24).

5.5.2 Rotated laplacian operator (*ln_dynldf_iso*=.true.)

A rotation of the lateral momentum diffusive operator is needed in several cases : for iso-neutral diffusion in z-coordinate (*ln_dynldf_iso*=.true.) and for either iso-neutral (*ln_dynldf_iso*=.true.) or geopotential (*ln_dynldf_hor*=.true.) diffusion in s-coordinate. In the partial step case, coordinates are horizontal excepted at the deepest level and no rotation is performed when *ln_dynldf_hor*=.true.. The diffusive operator is defined simply as the divergence of down gradient momentum fluxes on each momentum component. It must be emphasized that this formulation ignores constraints on the stress tensor such as symmetry. The resulting discrete representation is :

$$
\begin{aligned}
D_u^{l\mathbf{U}} = & \frac{1}{e_{1u}\, e_{2u}\, e_{3u}} \\
& \left\{ \quad \delta_{i+1/2}\left[A_T^{lm} \left(\frac{e_{2T}\; e_{3T}}{e_{1T}}\, \delta_i[u] - e_{2T}\; r_{1T}\, \overline{\overline{\delta_{k+1/2}[u]}}^{\,i,\,k} \right) \right] \right. \\
& \quad + \; \delta_j \left[A_f^{lm} \left(\frac{e_{1f}\, e_{3f}}{e_{2f}}\, \delta_{j+1/2}[u] - e_{1f}\, r_{2f}\, \overline{\overline{\delta_{k+1/2}[u]}}^{\,j+1/2,\,k} \right) \right] \\
& \quad + \; \delta_k \left[A_{uw}^{lm} \left(-e_{2u}\, r_{1uw}\, \overline{\overline{\delta_{i+1/2}[u]}}^{\,i+1/2,\,k+1/2} \right. \right. \\
& \qquad\qquad - e_{1u}\, r_{2uw}\, \overline{\overline{\delta_{j+1/2}[u]}}^{\,j,\,k+1/2} \\
& \qquad\qquad \left. \left. \left. + \frac{e_{1u}\, e_{2u}}{e_{3uw}} \left(r_{1uw}^2 + r_{2uw}^2 \right)\, \delta_{k+1/2}[u] \right) \right] \quad \right\}
\end{aligned}
\tag{5.25}
$$

$$
\begin{aligned}
D_v^{l\mathbf{V}} = & \frac{1}{e_{1v}\, e_{2v}\, e_{3v}} \\
& \left\{ \quad \delta_{i+1/2}\left[A_f^{lm} \left(\frac{e_{2f}\; e_{3f}}{e_{1f}}\, \delta_{i+1/2}[v] - e_{2f}\; r_{1f}\, \overline{\overline{\delta_{k+1/2}[v]}}^{\,i+1/2,\,k} \right) \right] \right. \\
& \quad + \; \delta_j \left[A_T^{lm} \left(\frac{e_{1T}\, e_{3T}}{e_{2T}}\, \delta_j[v] - e_{1T}\, r_{2T}\, \overline{\overline{\delta_{k+1/2}[v]}}^{\,j,\,k} \right) \right] \\
& \quad + \; \delta_k \left[A_{vw}^{lm} \left(-e_{2v}\, r_{1vw}\, \overline{\overline{\delta_{i+1/2}[v]}}^{\,i+1/2,\,k+1/2} \right. \right. \\
& \qquad\qquad - e_{1v}\, r_{2vw}\, \overline{\overline{\delta_{j+1/2}[v]}}^{\,j+1/2,\,k+1/2} \\
& \qquad\qquad \left. \left. \left. + \frac{e_{1v}\, e_{2v}}{e_{3vw}} \left(r_{1vw}^2 + r_{2vw}^2 \right)\, \delta_{k+1/2}[v] \right) \right] \quad \right\}
\end{aligned}
$$

where r_1 and r_2 are the slopes between the surface along which the diffusive operator acts and the surface of computation (z- or s-surfaces). The way these slopes are evaluated is given in the lateral physics chapter (Chap.8).

5.5.3 Iso-level bilaplacian operator (*ln_dynldf_bilap*=.true.)

The lateral fourth order operator formulation on momentum is obtained by applying (5.24) twice. It requires an additional assumption on boundary conditions : the first derivative term normal to the coast depends on the free or no-slip lateral boundary conditions chosen, while the third derivative terms normal to the coast are set to zero (see Chap.7).

5.6 Vertical diffusion term (*dynzdf.F90*)

```
!----------------------------------------------------------------------
```

```
&namzdf        !   vertical physics
!----------------------------------------------------------------------
   avm0        =   1.2e-4  !  vertical eddy viscosity   [m2/s]          (background Kz if not "key_zdfcst")
   avt0        =   1.2e-5  !  vertical eddy diffusivity [m2/s]          (background Kz if not "key_zdfcst")
   ln_zdfnpc   = .false.   !  convection: Non-Penetrative algorithm (T) or not (F)
   ln_zdfevd   = .true.    !  convection: enhanced vertical diffusion (T) or not (F)
   avevd       = 100.      !  vertical coefficient for enhanced diffusion scheme [m2/s]
   n_evdm      =   1       !  enhanced mixing apply on tracer (=0) or on tracer and momentum (=1)
   ln_zdfexp   =  .false.  !  split explicit (T) or implicit (F) time stepping
   n_zdfexp    =   3       !  number of sub-timestep for ln_zdfexp=T
/
```

The large vertical diffusion coefficient found in the surface mixed layer together with high vertical resolution implies that in the case of explicit time stepping there would be too restrictive a constraint on the time step. Two time stepping schemes can be used for the vertical diffusion term : (a) a forward time differencing scheme (*ln_zdfexp*=.true.) using a time splitting technique (*n_zdfexp* > 1) or (b) a backward (or implicit) time differencing scheme (*ln_zdfexp*=.false.) (see §3.4). Note that namelist variables *ln_zdfexp* and *n_zdfexp* apply to both tracers and dynamics.

The formulation of the vertical subgrid scale physics is the same whatever the vertical coordinate is. The vertical diffusion operators given by (2.43) take the following semi-discrete space form :

$$\left\{ \begin{aligned} D_u^{vm} &\equiv \frac{1}{e_{3u}} \; \delta_k \left[\frac{A_{uw}^{vm}}{e_{3uw}} \; \delta_{k+1/2}[\,u\,] \right] \\ \\ D_v^{vm} &\equiv \frac{1}{e_{3v}} \; \delta_k \left[\frac{A_{vw}^{vm}}{e_{3vw}} \; \delta_{k+1/2}[\,v\,] \right] \end{aligned} \right. \tag{5.26}$$

where A_{uw}^{vm} and A_{vw}^{vm} are the vertical eddy viscosity and diffusivity coefficients. The way these coefficients are evaluated depends on the vertical physics used (see §9).

The surface boundary condition on momentum is given by the stress exerted by the wind. At the surface, the momentum fluxes are prescribed as the boundary condition on the vertical turbulent momentum fluxes,

$$\left. \left(\frac{A^{vm}}{e_3} \; \frac{\partial \mathbf{U}_h}{\partial k} \right) \right|_{z=1} = \frac{1}{\rho_o} \binom{\tau_u}{\tau_v} \tag{5.27}$$

where (τ_u, τ_v) are the two components of the wind stress vector in the **(i,j)** coordinate system. The high mixing coefficients in the surface mixed layer ensure that the surface wind stress is distributed in the vertical over the mixed layer depth. If the vertical mixing coefficient is small (when no mixed layer scheme is used) the surface stress enters only the top model level, as a body force. The surface wind stress is calculated in the surface module routines (SBC, see Chap.6)

The turbulent flux of momentum at the bottom of the ocean is specified through a bottom friction parameterisation (see §9.4)

5.7 External Forcings

Besides the surface and bottom stresses (see the above section) which are introduced as boundary conditions on the vertical mixing, two other forcings enter the dynamical equations.

One is the effect of atmospheric pressure on the ocean dynamics (to be introduced later).

Another forcing term is the tidal potential, which will be introduced in the reference version soon.

5.8 Time evolution term (*dynnxt.F90*)

```
!-----------------------------------------------------------------------
&namdom        !   space and time domain (bathymetry, mesh, timestep)
!-----------------------------------------------------------------------
   ntopo       =    1      !  compute (=0) or read(=1) the bathymetry file
   e3zps_min   =    5.     !  the thickness of the partial step is set larger than the minimum
   e3zps_rat   =    0.1    !  of e3zps_min and e3zps_rat * e3t   (N.B. 0<e3zps_rat<1)
   nmsh        =    0      !  create (=1) a mesh file (coordinates, scale factors, masks) or not (=0)
   nacc        =    0      !  =1 acceleration of convergence method used, rdt < rdttra(k)
                           !  =0, no acceleration, rdt = rdttra
   atfp        =    0.1    !  asselin time filter parameter
   rdt         = 5760.     !  time step for the dynamics (and tracer if nacc=0)
   rdtmin      = 5760.     !  minimum time step on tracers (used if nacc=1)
   rdtmax      = 5760.     !  maximum time step on tracers (used if nacc=1)
   rdth        =  800.     !  depth variation of tracer time step  (used if nacc=1)
   rdtbt       =   90.     !  barotropic time step (for the split explicit algorithm) ("key_dynspg_ts")
   nclosea     =    0      !  = 0 no closed sea in the model domain
                           !  = 1 closed sea (Black Sea, Caspian Sea, Great US Lakes...)
/
```

The general framework for dynamics time stepping is a leap-frog scheme, $i.e.$ a three level centred time scheme associated with an Asselin time filter (cf. §3.4)

$$\begin{aligned} u^{t+\Delta t} &= u^{t-\Delta t} + 2\,\Delta t \ \mathrm{RHS}_u^t \\ u_f^t \quad &= u^t + \gamma \left[u_f^{t-\Delta t} - 2u^t + u^{t+\Delta t} \right] \end{aligned} \tag{5.28}$$

where RHS is the right hand side of the momentum equation, the subscript f denotes filtered values and γ is the Asselin coefficient. γ is initialized as *atfp* (namelist parameter). Its default value is *atfp* = 0.1.

Note that whith the filtered free surface, the update of the *next* velocities is done in the *dynsp_flt.F90* module, and only the swap of arrays and Asselin filtering is done in *dynnxt..F90*

5.9 Diagnostic variables (ζ, χ, w)

5.9.1 Horizontal divergence and relative vorticity (*divcur.F90*)

The vorticity is defined at an f-point ($i.e.$ corner point) as follows :

$$\zeta = \frac{1}{e_{1f}\,e_{2f}} \left(\ \delta_{i+1/2}\left[e_{2v}\ v\right] - \delta_{j+1/2}\left[e_{1u}\ u\right] \ \right) \tag{5.29}$$

The horizontal divergence is defined at a T-point. It is given by :

$$\chi = \frac{1}{e_{1T}\,e_{2T}\,e_{3T}} \left(\delta_i\left[e_{2u}\,e_{3u}\,u\right] + \delta_j\left[e_{1v}\,e_{3v}\,v\right] \right) \tag{5.30}$$

Note that in the z-coordinate with full step (**key_zco** is defined), $e_{3u} = e_{3v} = e_{3f}$ so that they cancel in (5.30).

Note also that whereas the vorticity have the same discrete expression in z- and s-coordinate, its physical meaning is not identical. ζ is a pseudo vorticity along s-surfaces (only pseudo because (u, v) are still defined along geopotential surfaces, but are no more necessary defined at the same depth).

The vorticity and divergence at the *before* step are used in the computation of the horizontal diffusion of momentum. Note that because they have been calculated prior to the Asselin filtering of the *before* velocities, the *before* vorticity and divergence arrays must be included in the restart file to ensure perfect restartability. The vorticity and divergence at the *now* time step are used for the computation of the nonlinear advection and of the vertical velocity respectively.

5.9.2 Vertical velocity (*wzvmod.F90*)

The vertical velocity is computed by an upward integration of the horizontal divergence from the bottom :

$$\left\{ \begin{aligned} \left. w \right|_{3/2} \quad &= 0 \\ \\ \left. w \right|_{k+1/2} &= \left. w \right|_{k+1/2} + e_{3t} \;\; \left. \chi \right|_k \end{aligned} \right. \tag{5.31}$$

With a free surface, the top vertical velocity is non-zero, due to the freshwater forcing and the variations of the free surface elevation. With a linear free surface or with a rigid lid, the upper boundary condition applies at a fixed level $z = 0$. Note that in the rigid-lid case (**key_dynspg_rl** is defined), the surface boundary condition ($\left. w \right|_{\text{surface}} = 0$) is automatically achieved at least at computer accuracy, due to the the way the surface pressure gradient is expressed in discrete form (Appendix C).

Note also that whereas the vertical velocity has the same discrete expression in z- and s-coordinate, its physical meaning is not the same : in the second case, w is the velocity normal to the s-surfaces.

With the variable volume option, the calculation of the vertical velocity is modified (see **?**, report available on the *NEMO* web site).

Surface Boundary Condition (SBC)

Contents

```
!-----------------------------------------------------------------------
&namsbc        !   Surface Boundary Condition (surface module)
!-----------------------------------------------------------------------
   nn_fsbc     = 5         !  frequency of surface boundary condition computation
                           !               (= the frequency of sea-ice model call)
   ln_ana      = .false.   !  analytical formulation (T => fill namsbc_ana )
   ln_flx      = .false.   !  flux formulation       (T => fill namsbc_flx )
   ln_blk_clio = .true.    !  CLIO bulk formulation  (T => fill namsbc_clio)
   ln_blk_core = .false.   !  CORE bulk formulation  (T => fill namsbc_core)
   ln_cpl      = .false.   !  Coupled formulation    (T => fill namsbc_cpl )
   nn_ice      = 2         !  =0 no ice boundary condition   ,
                           !  =1 use observed ice-cover      ,
                           !  =2 ice-model used                         ("key_lim3" or "key_lim2)
   nn_ico_cpl  = 0         !  ice-ocean coupling : =0 each nn_fsbc
                           !                       =1 stress recomputed each ocean time step ("key_lim3")
                           !                       =2 combination of 0 and 1 cases           ("key_lim3")
   ln_dm2dc    = .false.   !  daily mean to diurnal cycle short wave (qsr)
   ln_rnf      = .true.    !  runoffs (T => fill namsbc_rnf)
   ln_ssr      = .true.    !  Sea Surface Restoring on T and/or S (T => fill namsbc_ssr)
   nn_fwb      = 0         !  FreshWater Budget: =0 unchecked                              ,
                           !                     =1 global mean of e-p-r set to zero at each nn_fsbc time step   ,
                           !                     =2 annual global mean of e-p-r set to zero
/
```

The ocean needs six fields as surface boundary condition :

– the two components of the surface ocean stress $(\tau_u\ ,\ \tau_v)$
– the incoming solar and non solar heat fluxes $(Q_{ns}\ ,\ Q_{sr})$
– the surface freshwater budget $(\text{EMP},\ \text{EMP}_S)$

Four different ways to provide those six fields to the ocean are available which are controlled by namelist variables : an analytical formulation (*ln_ana*=true), a flux formulation (*ln_flx*=true), a bulk formulae formulation (CORE (*ln_core*=true) or CLIO (*ln_clio*=true) bulk formulae) and a coupled formulation (exchanges with a atmospheric model via the OASIS coupler) (*ln_cpl*=true). The frequency at which the six fields have to be updated is the *nf_sbc* namelist parameter. When the fields are supplied from data files (flux and bulk formulations), the input fields need not be supplied on the model grid. Instead a file of coordinates and weights can be supplied which maps the data from the supplied grid to the model points (so called ”Interpolation on the Fly”). In addition, the resulting fields can be further modified using several namelist options. These options control the rotation of vector components supplied relative to an east-north coordinate system onto the local grid directions in the model ; the addition of a surface restoring term to observed SST and/or SSS (*ln_ssr*=true) ; the modification of fluxes below ice-covered areas (using observed ice-cover or a sea-ice model) (*nn_ice*=0,1, 2 or 3) ; the addition of river runoffs (*ln_rnf*=true) ; the addition of a freshwater flux adjustment in order to avoid a mean sea-level drift (*nn_fwb*= 0, 1 or 2) ; and the transformation of the solar radiation (if provided as daily mean) into a diurnal cycle (*ln_dm2dc*=true).

In this chapter, we first discuss where the surface boundary condition appears in the model equations. Then we present the four ways of providing the surface boundary condition. Next the scheme for interpolation on the fly is described. Finally, the different options that further modify the fluxes applied to the ocean are discussed.

6.1 Surface boundary condition for the ocean

The surface ocean stress is the stress exerted by the wind and the sea-ice on the ocean. The two components of stress are assumed to be interpolated onto the ocean mesh, $i.e.$ resolved onto the model (**i**,**j**) direction at u- and v-points They are applied as a surface boundary condition of the computation of the momentum vertical mixing trend (*dynzdf.F90* module) :

$$\left(\frac{A^{vm}}{e_3} \; \frac{\partial \mathbf{U}_h}{\partial k} \right) \bigg|_{z=1} = \frac{1}{\rho_o} \begin{pmatrix} \tau_u \\ \tau_v \end{pmatrix} \tag{6.1}$$

where $(\tau_u, \; \tau_v) = (utau, vtau)$ are the two components of the wind stress vector in the $(\mathbf{i}, \mathbf{j})$ coordinate system.

The surface heat flux is decomposed into two parts, a non solar and a solar heat flux, Q_{ns} and Q_{sr}, respectively. The former is the non penetrative part of the heat flux ($i.e.$ the sum of sensible, latent and long wave heat fluxes). It is applied as a surface boundary condition trend of the first level temperature time evolution equation (*trasbc.F90* module).

$$\frac{\partial T}{\partial t} \equiv \cdots \; + \; \left. \frac{Q_{ns}}{\rho_o \; C_p \; e_{3T}} \right|_{k=1} \tag{6.2}$$

Q_{sr} is the penetrative part of the heat flux. It is applied as a 3D trends of the temperature equation (*traqsr.F90* module) when *ln_traqsr*=True.

$$\frac{\partial T}{\partial t} \equiv \cdots \; + \frac{Q_{sr}}{\rho_o C_p \, e_{3T}} \delta_k \left[I_w \right] \tag{6.3}$$

where I_w is a non-dimensional function that describes the way the light penetrates inside the water column. It is generally a sum of decreasing exponentials (see §4.4.2).

The surface freshwater budget is provided by fields : EMP and EMP$_S$ which may or may not be identical. Indeed, a surface freshwater flux has two effects : it changes the volume of the ocean and it changes the surface concentration of salt (and other tracers). Therefore it appears in the sea surface height as a volume flux, EMP (*dynspg_xxx* modules), and in the salinity time evolution equations as a concentration/dilution effect, EMP$_S$ (*trasbc.F90* module).

$$\begin{aligned} \frac{\partial \eta}{\partial t} &\equiv \cdots \; + \; \text{EMP} \\ \\ \frac{\partial S}{\partial t} &\equiv \cdots \; + \left. \frac{\text{EMP}_S \; S}{e_{3T}} \right|_{k=1} \end{aligned} \tag{6.4}$$

In the real ocean, EMP$=$EMP$_S$ and the ocean salt content is conserved, but it exist several numerical reasons why this equality should be broken. For example :

When the rigid-lid assumption is made, the ocean volume becomes constant and thus, EMP$=$0, not EMP$_S$.

When the ocean is coupled to a sea-ice model, the water exchanged between ice and ocean is slightly salty (mean sea-ice salinity is ∼*4 psu*). In this case, EMP$_S$ take into

Variable description	Model variable	Units	point
i-component of the surface current	ssu_m	$m.s^{-1}$	U
j-component of the surface current	ssv_m	$m.s^{-1}$	V
Sea surface temperature	sst_m	$^\circ K$	T
Sea surface salinity	sss_m	psu	T

TAB. 6.1 – Ocean variables provided by the ocean to the surface module (SBC). The variable are averaged over nf_sbc time step, $i.e.$ the frequency of computation of surface fluxes.

account both concentration/dilution effect associated with freezing/melting and the salt flux between ice and ocean, while EMP is only the volume flux. In addition, in the current version of *NEMO* , the sea-ice is assumed to be above the ocean. Freezing/melting does not change the ocean volume (no impact on EMP) but it modifies the SSS.

Note that SST can also be modified by a freshwater flux. Precipitation (in particular solid precipitation) may have a temperature significantly different from the SST. Due to the lack of information about the temperature of precipitation, we assume it is equal to the SST. Therefore, no concentration/dilution term appears in the temperature equation. It has to be emphasised that this absence does not mean that there is no heat flux associated with precipitation ! Precipitation can change the ocean volume and thus the ocean heat content. It is therefore associated with a heat flux (not yet diagnosed in the model) [**?**]).

The ocean model provides the surface currents, temperature and salinity averaged over *nf_sbc* time-step (6.1).The computation of the mean is done in *sbcmod.F90* module.

6.2 Analytical formulation (*sbcana.F90* module)

```
!-----------------------------------------------------------------------
&namsbc_ana    !   analytical surface boundary condition
!-----------------------------------------------------------------------
   nn_tau000   =   0       !  gently increase the stress over the first ntau_rst time-steps
   rn_utau0    =   0.5     !  uniform value for the i-stress
   rn_vtau0    =   0.e0    !  uniform value for the j-stress
   rn_q0       =   0.e0    !  uniform value for the total heat flux
   rn_qsr0     =   0.e0    !  uniform value for the solar radiation
   rn_emp0     =   0.e0    !  uniform value for the freswater budget (E-P)
/
```

The analytical formulation of the surface boundary condition is the default scheme. In this case, all the six fluxes needed by the ocean are assumed to be uniform in space. They take constant values given in the namelist namsbc_ana by the variables *rn_utau0*, *rn_vtau0*, *rn_qns0*, *rn_qsr0*, and *rn_emp0* (EMP=EMP$_S$). The runoff is set to zero. In addition, the wind is allowed to reach its nominal value within a given number of time steps (*nn_tau000*).

If a user wants to apply a different analytical forcing, the *sbcana.F90* module can be modified to use another scheme. As an example, the *sbc_ana_gyre.F90* routine provides the analytical forcing for the GYRE configuration (see GYRE configuration manual, in preparation).

6.3 Flux formulation (*sbcflx.F90* module)

```
!-----------------------------------------------------------------------
&namsbc_flx    !   surface boundary condition : flux formulation
!-----------------------------------------------------------------------
!              !      file name     ! frequency (hours) !  variable  ! time interpol. !  clim   ! 'yearly' or !
!              !                    !  (if <0  months)  !    name    !    (logical)   !  (T/F)  !  'monthly'  !
   sn_utau      =      'utau.nc'     ,        24.        ,  'utau'  ,      .false.       ,    0   ,     0    , ''       , ''
   sn_vtau      =      'vtau.nc'     ,        24.        ,  'vtau'  ,      .false.       ,    0   ,     0    , ''       , ''
   sn_qtot      =      'qtot.nc'     ,        24.        ,  'qtot'  ,      .false.       ,    0   ,     0    , ''       , ''
   sn_qsr       =      'qsr.nc'      ,        24.        ,  'qsr'   ,      .false.       ,    0   ,     0    , ''       , ''
   sn_emp       =      'emp.nc'      ,        24.        ,  'emp'   ,      .false.       ,    0   ,     0    , ''       , ''
!
   cn_dir      = './'      !  root directory for the location of the flux files
/
```

In the flux formulation (*ln_flx*=true), the surface boundary condition fields are directly read from input files. The user has to define in the namelist namsbc_flx the name of the file, the name of the variable read in the file, the time frequency at which it is given (in hours), and a logical setting whether a time interpolation to the model time step is required for this field). (fld_i namelist structure).

Caution : when the frequency is set to –12, the data are monthly values. These are assumed to be climatological values, so time interpolation between December the 15^{th} and January the 15^{th} is done using records 12 and 1

When higher frequency is set and time interpolation is demanded, the model will try to read the last (first) record of previous (next) year in a file having the same name but a suffix _prev_year (_next_year) being added (e.g. "_1989"). These files must only contain a single record. If they don't exist, the model assumes that the last record of the previous year is equal to the first record of the current year, and similarly, that the first record of the next year is equal to the last record of the current year. This will cause the forcing to remain constant over the first and last half fld_frequ hours.

Note that in general, a flux formulation is used in associated with a restoring term to observed SST and/or SSS. See §6.7.2 for its specification.

6.4 Bulk formulation (*sbcblk_core.F90* or *sbcblk_clio.F90* module)

In the bulk formulation, the surface boundary condition fields are computed using bulk formulae and atmospheric fields and ocean (and ice) variables.

The atmospheric fields used depend on the bulk formulae used. Two bulk formulations are available : the CORE and CLIO bulk formulea. The choice is made by setting to true one of the following namelist variable : *ln_core* and *ln_clio*.

Note : in forced mode, when a sea-ice model is used, a bulk formulation have to be used. Therefore the two bulk formulea provided include the computation of the fluxes over both an ocean and an ice surface.

6.4.1 CORE Bulk formulea (*ln_core*=true, *sbcblk_core.F90*)

```
!-----------------------------------------------------------------------
&namsbc_core   !   namsbc_core  CORE bulk formulea
!-----------------------------------------------------------------------
!              !   file name        ! frequency (hours) ! variable   ! time interpol. !  clim   ! 'yearly' or !
!              !                    !  (if <0  months)  !   name     !    (logical)   !  (T/F)  !  'monthly'  !
   sn_wndi      =    'u10'           ,       24.         , 'U_10_MOD' ,     .false.        ,    1   ,    0     , ''       , ''
   sn_wndj      =    'v10'           ,       24.         , 'V_10_MOD' ,     .false.        ,    1   ,    0     , ''       , ''
   sn_qsr      =    'rad'          ,       24.        , 'SWDN_MOD' ,     .false.       ,    1   ,    0     , ''       , ''
   sn_qlw      =    'rad'          ,       24.        , 'LWDN_MOD' ,     .false.       ,    1   ,    0     , ''       , ''
   sn_tair     =    't10'          ,       24.        , 'T_10_MOD' ,     .false.       ,    1   ,    0     , ''       , ''
   sn_humi     =    'q10'          ,       24.        , 'Q_10_MOD' ,     .false.       ,    1   ,    0     , ''       , ''
   sn_prec     =    'precip'       ,      -12.        , 'RAIN'     ,     .false.       ,    1   ,    0     , ''       , ''
   sn_snow     =    'precip'       ,      -12.        , 'SNOW'     ,     .false.       ,    1   ,    0     , ''       , ''
!
   cn_dir      = './'      !  root directory for the location of the bulk files
   ln_2m       = .false.   !  air temperature and humidity referenced at 2m (T) instead 10m (F)
   alpha_precip= 1.        !  multiplicative factor for precipitation (total & snow)
/
```

The CORE bulk formulae have been developed by **?**. They have been designed to handle the CORE forcing, a mixture of NCEP reanalysis and satellite data. They use an inertial dissipative method to compute the turbulent transfer coefficients (momentum, sensible heat and evaporation) from the 10 metre wind speed, air temperature and specific humidity.

Note that substituting ERA40 to NCEP reanalysis fields does not require changes in the bulk formulea themself.

The required 8 input fields are :

Variable desciption	Model variable	Units	point
i-component of the 10m air velocity	utau	$m.s^{-1}$	T
j-component of the 10m air velocity	vtau	$m.s^{-1}$	T
10m air temperature	tair	$^\circ K$	T
Specific humidity	humi	%	T
Incoming long wave radiation	qlw	$W.m^{-2}$	T
Incoming short wave radiation	qsr	$W.m^{-2}$	T
Total precipitation (liquid + solid)	precip	$Kg.m^{-2}.s^{-1}$	T
Solid precipitation	snow	$Kg.m^{-2}.s^{-1}$	T

Note that the air velocity is provided at a tracer ocean point, not at a velocity ocean point (u- and v-points). It is simpler and faster (less fields to be read), but it is not the recommended method when the ocean grid size is the same or larger than the one of the input atmospheric fields.

6.4.2 CLIO Bulk formulea (*ln_clio*=true, *sbcblk_clio.F90*)

```
!-----------------------------------------------------------------------
&namsbc_clio             CLIO bulk formulea
!-----------------------------------------------------------------------
!              !      file name     ! frequency (hours) !  variable  ! time interpol. !  clim   ! 'yearly' or !
!              !                    !  (if <0  months)  !    name    !    (logical)   !  (T/F)  !  'monthly'  !
   sn_utau      =    'taux_1m'      ,      -12.         , 'sozotaux' ,     .false.       ,    1   ,    0     , ''       , ''
   sn_vtau      =    'tauy_1m'      ,      -12.         , 'sometauy' ,     .false.       ,    1   ,    0     , ''       , ''
   sn_wndm      =    'flx'          ,      -12.         , 'socliowi' ,     .false.       ,    1   ,    0     , ''       , ''
   sn_tair      =    'flx'          ,      -12.         , 'socliot1' ,     .false.       ,    1   ,    0     , ''       , ''
   sn_humi      =    'flx'          ,      -12.         , 'socliohu' ,     .false.       ,    1   ,    0     , ''       , ''
   sn_ccov      =    'flx'          ,      -12.         , 'socliocl' ,     .false.       ,    1   ,    0     , ''       , ''
   sn_prec      =    'flx'          ,      -12.         , 'socliopl' ,     .false.       ,    1   ,    0     , ''       , ''

!
   cn_dir      = './'      !  root directory for the location of the bulk files are
/
```

The CLIO bulk formulae were developed several years ago for the Louvain-la-neuve coupled ice-ocean model (CLIO, **?**). They are simpler bulk formulae. They assume the stress to be known and compute the radiative fluxes from a climatological cloud cover.

The required 7 input fields are :

Variable desciption	Model variable	Units	point
i-component of the ocean stress	utau	$N.m^{-2}$	U
j-component of the ocean stress	vtau	$N.m^{-2}$	V
Wind speed module	vatm	$m.s^{-1}$	T
10m air temperature	tair	$^\circ K$	T
Specific humidity	humi	%	T
Cloud cover		%	T
Total precipitation (liquid + solid)	precip	$Kg.m^{-2}.s^{-1}$	T
Solid precipitation	snow	$Kg.m^{-2}.s^{-1}$	T

As for the flux formulation, information about the input data required by the model is provided in the namsbc_blk_core or namsbc_blk_clio namelist (via the structure fld_i). The first and last record assumption is also made (see §6.3)

6.5 Coupled formulation (*sbccpl.F90* module)

```
!-----------------------------------------------------------------------
&namsbc_cpl    !   coupled ocean/atmosphere model                        ("key_coupled")
!-----------------------------------------------------------------------
! SEND
cn_snd_temperature= 'weighted oce and ice'  ! 'oce only' 'weighted oce and ice' 'mixed oce-ice'
cn_snd_albedo     = 'weighted ice'          ! 'none' 'weighted ice' 'mixed oce-ice'
cn_snd_thickness  = 'none'                  ! 'none' 'weighted ice and snow'
cn_snd_crt_nature = 'none'                  ! 'none' 'oce only' 'weighted oce and ice' 'mixed oce-ice'
cn_snd_crt_refere = 'spherical'             ! 'spherical' 'cartesian'
cn_snd_crt_orient = 'eastward-northward'    ! 'eastward-northward' or 'local grid'
cn_snd_crt_grid   = 'T'                     ! 'T'
! RECEIVE
cn_rcv_w10m       = 'coupled'               ! 'none' 'coupled'
cn_rcv_tau_nature = 'oce only'              ! 'oce only' 'oce and ice' 'mixed oce-ice'
cn_rcv_tau_refere = 'cartesian'             ! 'spherical' 'cartesian'
cn_rcv_tau_orient = 'eastward-northward'    ! 'eastward-northward' or 'local grid'
cn_rcv_tau_grid   = 'U,V'                   ! 'T' 'U,V' 'U,V,F' 'U,V,I' 'T,F' 'T,I' 'T,U,V'
cn_rcv_dqnsdt     = 'coupled'               ! 'none' 'coupled'
cn_rcv_qsr        = 'oce and ice'           ! 'conservative' 'oce and ice' 'mixed oce-ice'
cn_rcv_qns        = 'oce and ice'           ! 'conservative' 'oce and ice' 'mixed oce-ice'
cn_rcv_emp        = 'conservative'          ! 'conservative' 'oce and ice' 'mixed oce-ice'
cn_rcv_rnf        = 'coupled'               ! 'coupled' 'climato' 'mixed'
cn_rcv_cal        = 'coupled'               ! 'none' 'coupled'
/
```

In the coupled formulation of the surface boundary condition, the fluxes are provided by the OASIS coupler at each *nf_cpl* time-step, while sea and ice surface temperature, ocean and ice albedo, and ocean currents are sent to the atmospheric component.

The generalised coupled interface is under development. It should be available in summer 2008. It will include the ocean interface for most of the European atmospheric GCM (ARPEGE, ECHAM, ECMWF, HadAM, LMDz).

6.6 Interpolation on the Fly

Interpolation on the Fly allows the user to supply input files required for the surface forcing on grids other than the model grid. To do this he or she must supply, in addition to the source data file, a file of weights to be used to interpolate from the data grid to the model grid. The original development of this code used the SCRIP package (freely available under a copyright agreement from http ://climate.lanl.gov/Software/SCRIP). In principle, any package can be used to generate the weights, but the variables in the input weights file must have the same names and meanings as assumed by the model. Two methods are currently available : bilinear and bicubic interpolation.

6.6.1 Bilinear Interpolation

The input weights file in this case has two sets of variables : src01, src02, src03, src04 and wgt01, wgt02, wgt03, wgt04. The ”src” variables correspond to the point in the input grid to which the weight ”wgt” is to be applied. Each src value is an integer corresponding to the index of a point in the input grid when written as a one dimensional array. For example, for an input grid of size 5x10, point (3,2) is referenced as point 8, since (2-1)*5+3=8. There are four of each variable because bilinear interpolation uses the four points defining the grid box containing the point to be interpolated. All of these arrays are on the model grid, so that values src01(i,j) and wgt01(i,j) are used to generate a value for point (i,j) in the model.

Symbolically, the algorithm used is :

$$f_m(i,j) = f_m(i,j) + \sum_{k=1}^{4} wgt(k) f(idx(src(k))) \tag{6.5}$$

where function idx() transforms a one dimensional index src(k) into a two dimensional index, and wgt(1) corresponds to variable ”wgt01” for example.

6.6.2 Bicubic Interpolation

Again there are two sets of variables : ”src” and ”wgt”. But in this case there are 16 of each. The symbolic algorithm used to calculate values on the model grid is now :

$$
\begin{aligned}
f_m(i,j) = f_m(i,j) &+ \sum_{k=1}^{4} wgt(k) f(idx(src(k))) \\
&+ \sum_{k=5}^{8} wgt(k) \left.\frac{\partial f}{\partial i}\right|_{idx(src(k))} \\
&+ \sum_{k=9}^{12} wgt(k) \left.\frac{\partial f}{\partial j}\right|_{idx(src(k))} \\
&+ \sum_{k=13}^{16} wgt(k) \left.\frac{\partial^2 f}{\partial i \partial j}\right|_{idx(src(k))}
\end{aligned}
$$

The gradients here are taken with respect to the horizontal indices and not distances since the spatial dependency has been absorbed into the weights.

6.6.3 Implementation

To activate this option, a non-empty string should be supplied in the weights filename column of the relevant namelist; if this is left as an empty string no action is taken. In the model, weights files are read in and stored in a structured type (WGT) in the fldread module, as and when they are first required. This initialisation procedure tries to determine whether the input data grid should be treated as cyclical or not. (In fact this only matters when bicubic interpolation is required.) To do this the model looks in the input data file (i.e. the data to which the weights are to be applied) for a variable with name ”nav_lon” or ”lon”. If found, it checks the difference between the first and last values of longitude along a single row. If the absolute value of this difference is close to 360 degrees or less than twice the maximum spacing from 360 degrees, the grid is assumed to be cyclical, and the difference determines whether the first column is a repeat of the last one or not. If neither ”nav_lon” or ”lon” can be found, the model resorts to looking at the first and last columns of data. If the sum of the absolute values of the differences between the columns is very small, then the grid is assumed to be cyclical with coincident first and last columns. If both of these tests fail, the grid is assumed not to be cyclical.

Next the routine reads in the weights. Bicubic interpolation is assumed if it finds a variable with name ”src05”, otherwise bilinear interpolation is used. The WGT structure includes dynamic arrays both for the storage of the weights (on the model grid), and when required, for reading in the variable to be interpolated (on the input data grid). The size of the input data array is determined by examining the values in the ”src” arrays to find the minimum and maximum i and j values required. Since bicubic interpolation requires the calculation of gradients at each point on the grid, the corresponding arrays are dimensioned with a halo of width one grid point all the way around. When the array of points from the data file is adjacent to an edge of the data grid, the halo is either a copy of the row/column next to it (non-cyclical case), or is a copy of one from the first two rows/columns on the opposite side of the grid (cyclical case with coincident end rows/columns, or cyclical case with non-coincident end rows/columns).

6.6.4 Limitations

Input data grids must be logically rectangular.

This code is not guaranteed to produce positive definite answers from positive definite inputs.

The cyclic condition is only applied on left and right columns, and not to top and bottom rows.

The gradients across the ends of a cyclical grid assume that the grid spacing between the two columns involved are consistent with the weights used.

Neither interpolation scheme is conservative. (There is a conservative scheme available in SCRIP, but this has not been implemented.)

6.6.5 Utilities

A set of utilities to create a weights file for a rectilinear input grid is available.

6.7 Miscellaneous options

6.7.1 Rotation of vector pairs onto the model grid directions

When using a flux (*ln_flx*=true) or bulk (*ln_clio*=true or *ln_core*=true) formulation, pairs of vector components can be rotated from east-north directions onto the local grid directions. This is particularly useful when interpolation on the fly is used since here any vectors are likely to be defined relative to a rectilinear grid. To activate this option a non-empty string is supplied in the rotation pair column of the relevant namelist. The eastward component must start with ”U” and the northward component with ”V”. The remaining characters in the strings are used to identify which pair of components go together. So for example, strings ”U1” and ”V1” next to ”utau” and ”vtau” would pair the wind stress components together and rotate them on to the model grid directions ; ”U2” and ”V2” could be used against a second pair of components, and so on. The extra characters used in the strings are arbitrary. The rot_rep routine from the *geo2ocean.F90* module is used to perform the rotation.

6.7.2 Surface restoring to observed SST and/or SSS (*sbcssr.F90*)

```
!-----------------------------------------------------------------------
&namsbc_ssr    !   surface boundary condition : sea surface restoring
!-----------------------------------------------------------------------
!              !     file name      ! frequency (hours) ! variable   ! time interpol. !  clim   ! ’yearly’ or !
!              !                    !  (if <0  months)  !   name     !    (logical)   !  (T/F)  !  ’monthly’  !
   sn_sst       =  ’sst_data.nc’     ,        24.       ,  ’sst’     ,      .false.      ,    0   ,    0
   sn_sss       =  ’sss_data.nc’     ,       -12.       ,  ’sss’     ,      .true.       ,    0   ,    0
```

```
!
   cn_dir      = './'      !  root directory for the location of the runoff files
   nn_sstr     =    0      !  add a retroaction term in the surface heat       flux (=1) or not (=0)
   nn_sssr     =    1      !  add a damping     term in the surface freshwater flux (=1) or not (=0)
   dqdt        =  -40.     !  magnitude of the retroaction on temperature   [W/m2/K]
   deds        =  -27.7    !  magnitude of the damping on salinity   [mm/day/psu]
/
```

In forced mode using a flux formulation (default option or **key_flx** defined), a feedback term *must* be added to the surface heat flux Q_{ns}^o :

$$Q_{ns} = Q_{ns}^o + \frac{dQ}{dT} \left(\left. T \right|_{k=1} - SST_{Obs} \right) \tag{6.6}$$

where SST is a sea surface temperature field (observed or climatological), T is the model surface layer temperature and $\frac{dQ}{dT}$ is a negative feedback coefficient usually taken equal to $-40\ W/m^2/K$. For a $50\ m$ mixed-layer depth, this value corresponds to a relaxation time scale of two months. This term ensures that if T perfectly matches the supplied SST, then Q is equal to Q_o.

In the fresh water budget, a feedback term can also be added. Converted into an equivalent freshwater flux, it takes the following expression :

$$EMP = EMP_o + \gamma_s^{-1} e_{3t} \frac{\left(\left. S \right|_{k=1} - SSS_{Obs} \right)}{\left. S \right|_{k=1}} \tag{6.7}$$

where EMP$_o$ is a net surface fresh water flux (observed, climatological or an atmospheric model product), *SSS$_{Obs}$* is a sea surface salinity (usually a time interpolation of the monthly mean Polar Hydrographic Climatology [**?**]), $\left. S \right|_{k=1}$ is the model surface layer salinity and γ_s is a negative feedback coefficient which is provided as a namelist parameter. Unlike heat flux, there is no physical justification for the feedback term in 6.7 as the atmosphere does not care about ocean surface salinity [**?**]. The SSS restoring term should be viewed as a flux correction on freshwater fluxes to reduce the uncertainties we have on the observed freshwater budget.

6.7.3 Handling of ice-covered area (*sbcice_...*)

The presence at the sea surface of an ice covered area modifies all the fluxes transmitted to the ocean. There are several way to handle sea-ice in the system depending on the value of the *nn_ice* namelist parameter.

nn_ice = 0 there will never be sea-ice in the computational domain. This is a typical namelist value used for tropical ocean domain. The surface fluxes are simply specified for an ice-free ocean. No specific things is done for sea-ice.

nn_ice = 1 sea-ice can exist in the computational domain, but no sea-ice model is used. An observed ice covered area is read in a file. Below this area, the SST is restored to the freezing point and the heat fluxes are set to $-4\ W/m^2$ ($-2\ W/m^2$) in the northern (southern) hemisphere. The associated modification of the freshwater fluxes are done in such a way that the change in buoyancy fluxes remains zero. This prevents deep convection to occur when trying to reach the freezing point (and so ice covered area condition) while the SSS is too large. This manner of managing sea-ice area, just by using si IF case, is usually referred as the *ice-if* model. It can be found in the *sbcice_if.F90* module.

nn_ice = 2 or more A full sea ice model is used. This model computes the ice-ocean fluxes, that are combined with the air-sea fluxes using the ice fraction of each model cell to provide the surface ocean fluxes. Note that the activation of a sea-ice model is is done by defining a CPP key (**key_lim2** or **key_lim3**). The activation automatically ovewrite the read value of nn_ice to its appropriate value ($i.e.$ 2 for LIM-2 and 3 for LIM-3).

6.7.4 Addition of river runoffs (*sbcrnf.F90*)

```
!-----------------------------------------------------------------------
&namsbc_rnf    !   runoffs namelist surface boundary condition
!-----------------------------------------------------------------------
!              !   file name        ! frequency (hours) ! variable   ! time interpolation !  clim  ! starting !
!              !                    !   (if <0  months) !   name     !     (logical)      !  (0/1) !  record  !
   sn_rnf     = 'runoff_1m_nomask.nc' ,      -12.        ,  'sorunoff',      .true.        ,    1   ,    0     , ''       , ''
   sn_cnf     = 'runoff_1m_nomask.nc' ,        0.        ,  'socoefr' ,      .false.       ,    1   ,    0     , ''       , ''
   sn_s_rnf   = 'runoffs'             ,  24              , 'rosaline' ,    .true.          , .true. ,  'yearly', ''       , ''
   sn_t_rnf   = 'runoffs'             ,  24              , 'rotemper' ,    .true.          , .true. ,  'yearly', ''       , ''
   sn_dep_rnf = 'runoffs'             ,   0              , 'rodepth'  ,    .false.         , .true. ,  'yearly', ''       , ''
!
   cn_dir       = './'       ! directory in which the model is executed
   ln_rnf_emp   =   .false. !  runoffs included into precipitation field (T) or into a file (F)
   ln_rnf_mouth =   .false. !  specific treatment at rivers mouths
   rn_hrnf      =   0.e0    !  depth over which enhanced vertical mixing is used
   rn_avt_rnf   =   1.e-3   !  value of the additional vertical mixing coef. [m2/s]
   rn_rfact     =   1.e0    !  multiplicative factor for runoff
   ln_rnf_depth =  .false.  !  read in depth information for runoff
   ln_rnf_temp  =  .false.  !  read in temperature information for runoff
   ln_rnf_sal   =  .false.  !  read in salinity information for runoff
/
```

It is convenient to introduce the river runoff in the model as a surface fresh water flux. This is the default option within NEMO, and there is then the option for the user to increase vertical mixing in the vicinity of the rivermouth.

However, this method is not very appropriate for coastal modelling. As such its also possible to specify, in a netcdf input file, the temperature and salinity of the river, along with the depth (in metres) which the river should be added to. This enables to river to be correctly added through all or some of the water column, instead of as a surface flux, and also means the temperature and salinity (for low salinity outflow) of the river impacts the surrounding ocean.

For the river temperature variable, -999 is the missing data value and this causes river temperature to be taken as the surface temperature at the river point. For the depth parameter a value of -1 means the river is added to the surface box only, and a value of -999 means the river is added through the entire water column.

Namelist options, *ln_rnf_depth*, *ln_rnf_sal* and *ln_rnf_temp* control whether the river attributes (depth, salinity and temperature) are read in and used. If these are set as false the river is added to the surface box only, assumed to be fresh (0psu), and/or taken as surface temperature respectively.

It is also possible for runnoff to be specified as a negative value for modelling flow through straits, ie, modelling the Baltic flow in and out of the north sea. When the flow is out of the domain there is no change in temperature and salinity, regardless of the namelist options used.

The runoff value and attributes are read in in sbcrnf. The mass/volume addition is added to the divergence term in sbc_rnf_div. The dilution effect of the river is automatically applied through the vertical tracer advection, and the direct flux of tracers into the domain is done in trasbc.

6.7.5 Freshwater budget control (*sbcfwb.F90*)

For global ocean simulation it can be useful to introduce a control of the mean sea level in order to prevent unrealistic drift of the sea surface height due to inaccuracy in the freshwater fluxes. In *NEMO* , two way of controlling the the freshwater budget.

***nn_fwb*=0** no control at all. The mean sea level is free to drift, and will certainly do so.

***nn_fwb*=1** global mean EMP set to zero at each model time step.

***nn_fwb*=2** freshwater budget is adjusted from the previous year annual mean budget which is read in the *EMPave_old.dat* file. As the model uses the Boussinesq approximation, the annual mean fresh water budget is simply evaluated from the change in the mean sea level at January the first and saved in the *EMPav.dat* file.

Lateral Boundary Condition (LBC)

Contents

7.1 Boundary Condition at the Coast (*shlat*)

```
!-----------------------------------------------------------------------
&namlbc        !   lateral momentum boundary condition
!-----------------------------------------------------------------------
   shlat       =    2.     !      shlat = 0 : free slip
                           !  0 < shlat < 2 : partial slip
                           !      shlat = 2 : no slip
                           !  2 < shlat     : strong slip
/
```

The discrete representation of a domain with complex boundaries (coastlines and bottom topography) leads to arrays that include large portions where a computation is not required as the model variables remain at zero. Nevertheless, vectorial supercomputers are far more efficient when computing over a whole array, and the readability of a code is greatly improved when boundary conditions are applied in an automatic way rather than by a specific computation before or after each computational loop. An efficient way to work over the whole domain while specifying the boundary conditions, is to use multiplication by mask arrays in the computation. A mask array is a matrix whose elements are 1 in the ocean domain and 0 elsewhere. A simple multiplication of a variable by its own mask ensures that it will remain zero over land areas. Since most of the boundary conditions consist of a zero flux across the solid boundaries, they can be simply applied by multiplying variables by the correct mask arrays, $i.e.$ the mask array of the grid point where the flux is evaluated. For example, the heat flux in the **i**-direction is evaluated at u-points. Evaluating this quantity as,

$$\frac{A^{lT}}{e_1}\frac{\partial T}{\partial i} \equiv \frac{A_u^{lT}}{e_{1u}}\;\delta_{i+1/2}\left[T\right]\quad mask_u \tag{7.1}$$

(where mask$_u$ is the mask array at a u-point) ensures that the heat flux is zero inside land and at the boundaries, since mask$_u$ is zero at solid boundaries which in this case are defined at u-points (normal velocity u remains zero at the coast) (Fig. 7.1).

For momentum the situation is a bit more complex as two boundary conditions must be provided along the coast (one each for the normal and tangential velocities). The boundary of the ocean in the C-grid is defined by the velocity-faces. For example, at a given T-level, the lateral boundary (a coastline or an intersection with the bottom topography) is made of segments joining f-points, and normal velocity points are located between two $f-$points (Fig. 7.1). The boundary condition on the normal velocity (no flux through solid boundaries) can thus be easily implemented using the mask system. The boundary condition on the tangential velocity requires a more specific treatment. This boundary condition influences the relative vorticity and momentum diffusive trends, and is required in order to compute the vorticity at the coast. Four different types of lateral boundary condition are available, controlled by the value of the *shlat* namelist parameter. (The value of the mask$_f$ array along the coastline is set equal to this parameter.) These are :

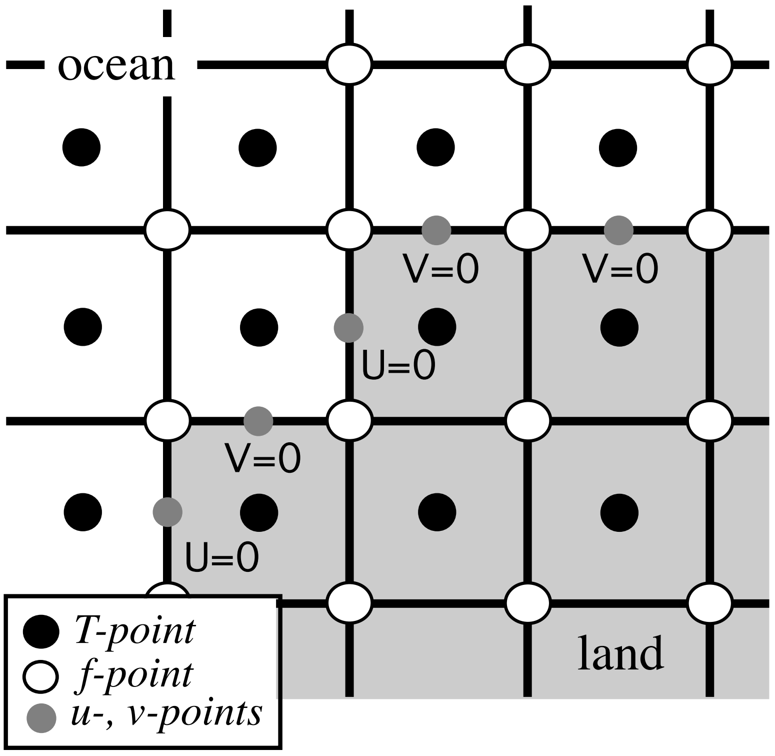


FIG. 7.1 – Lateral boundary (thick line) at T-level. The velocity normal to the boundary is set to zero.

free-slip boundary condition (*shlat*=0) : the tangential velocity at the coastline is equal to the offshore velocity, $i.e.$ the normal derivative of the tangential velocity is zero at the coast, so the vorticity : mask$_f$ array is set to zero inside the land and just at the coast (Fig. 7.1-a).

no-slip boundary condition (*shlat*=2) : the tangential velocity vanishes at the coastline. Assuming that the tangential velocity decreases linearly from the closest ocean velocity grid point to the coastline, the normal derivative is evaluated as if the velocities at the closest land velocity gridpoint and the closest ocean velocity gridpoint were of the same magnitude but in the opposite direction (Fig. 7.1-b). Therefore, the vorticity along the coastlines is given by :

$$\zeta \equiv 2 \left(\delta_{i+1/2} \left[e_{2v} v\right] - \delta_{j+1/2} \left[e_{1u} u\right]\right) / \left(e_{1f} e_{2f}\right) \ ,$$

where u and v are masked fields. Setting the mask$_f$ array to 2 along the coastline provides a vorticity field computed with the no-slip boundary condition, simply by

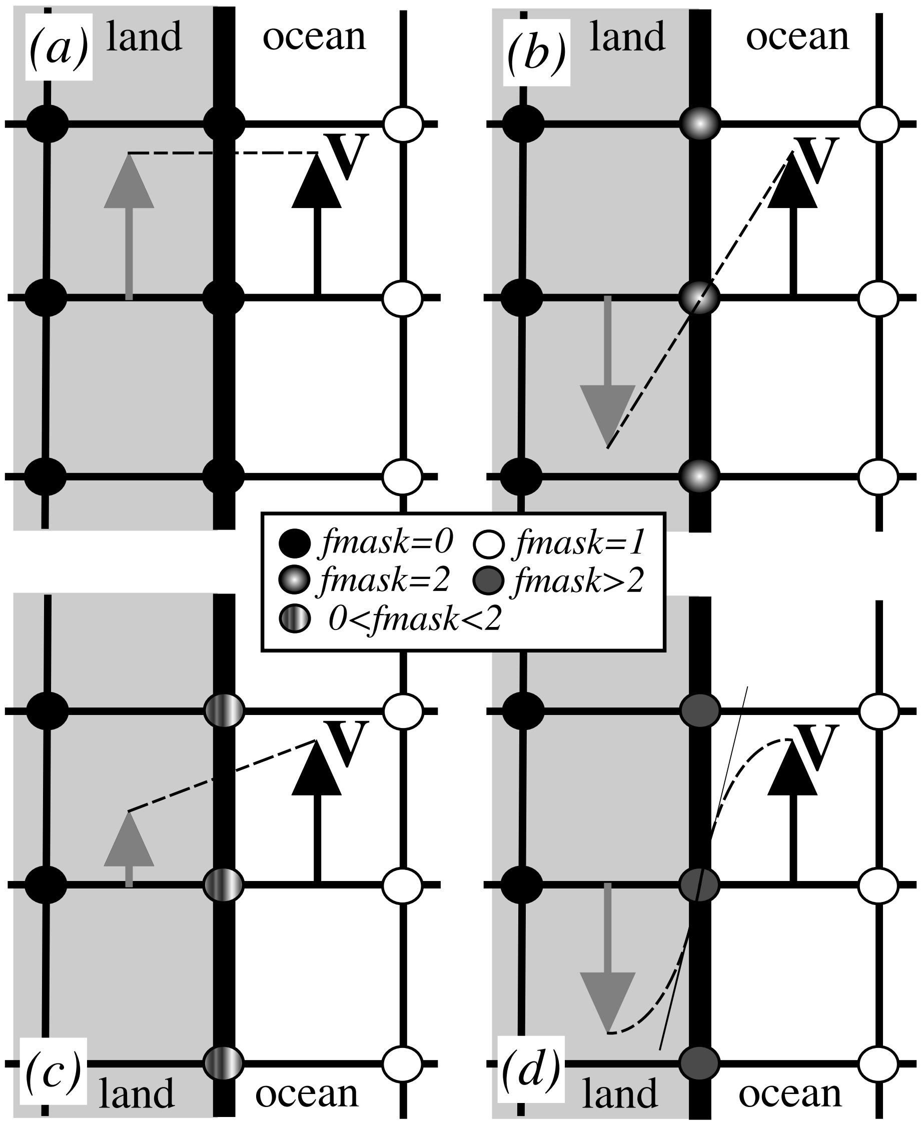


FIG. 7.2 – lateral boundary condition (a) free-slip ($shlat = 0$) ; (b) no-slip ($shlat = 2$) ; (c) "partial" free-slip ($0 < shlat < 2$) and (d) "strong" no-slip ($2 < shlat$). Implied "ghost" velocity inside land area is display in grey.

multiplying it by the mask_f :

$$\zeta \equiv \frac{1}{e_{1f}\, e_{2f}} \left(\delta_{i+1/2} \left[e_{2v}\, v \right] - \delta_{j+1/2} \left[e_{1u}\, u \right] \right) \; \text{mask}_f \tag{7.2}$$

"partial" free-slip boundary condition (0<*shlat*<2) : the tangential velocity at the coastline is smaller than the offshore velocity, $i.e.$ there is a lateral friction but not strong enough to make the tangential velocity at the coast vanish (Fig. 7.1-c). This can be selected by providing a value of mask_f strictly inbetween 0 and 2.

"strong" no-slip boundary condition (2<*shlat*) : the viscous boundary layer is assumed to be smaller than half the grid size (Fig. 7.1-d). The friction is thus larger than in the no-slip case.

Note that when the bottom topography is entirely represented by the s-coor-dinates (pure s-coordinate), the lateral boundary condition on tangential velocity is of much less importance as it is only applied next to the coast where the minimum water depth can be quite shallow.

The alternative numerical implementation of the no-slip boundary conditions for an arbitrary coast line of **?** is also available through the **key_noslip_accurate** CPP key. It is based on a fourth order evaluation of the shear at the coast which, in turn, allows a true second order scheme in the interior of the domain ($i.e.$ the numerical boundary scheme simulates the truncation error of the numerical scheme used in the interior of the domain). **?** found that such a technique considerably improves the quality of the numerical solution. In *NEMO* , such spectacular improvements have not been found in the half-degree global ocean (ORCA05), but significant reductions of numerically induced coastal upwellings were found in an eddy resolving simulation of the Alboran Sea [**?**]. Nevertheless, since a no-slip boundary condition is not recommended in an eddy permitting or resolving simulation [**?**], the use of this option is also not recommended.

In practice, the no-slip accurate option changes the way the curl is evaluated at the coast (see *divcur.F90* module), and requires the nature of each coastline grid point (convex or concave corners, straight north-south or east-west coast) to be specified. This is performed in routine *dom_msk_nsa* in the *domask.F90* module.

7.2 Model Domain Boundary Condition (*jperio*)

At the model domain boundaries several choices are offered : closed, cyclic east-west, south symmetric across the equator, a north-fold, and combination closed-north fold or cyclic-north-fold. The north-fold boundary condition is associated with the 3-pole ORCA mesh.

7.2.1 Closed, cyclic, south symmetric (*jperio* = 0, 1 or 2)

The choice of closed, cyclic or symmetric model domain boundary condition is made by setting *jperio* to 0, 1 or 2 in file *par_oce.F90*. Each time such a boundary condition is

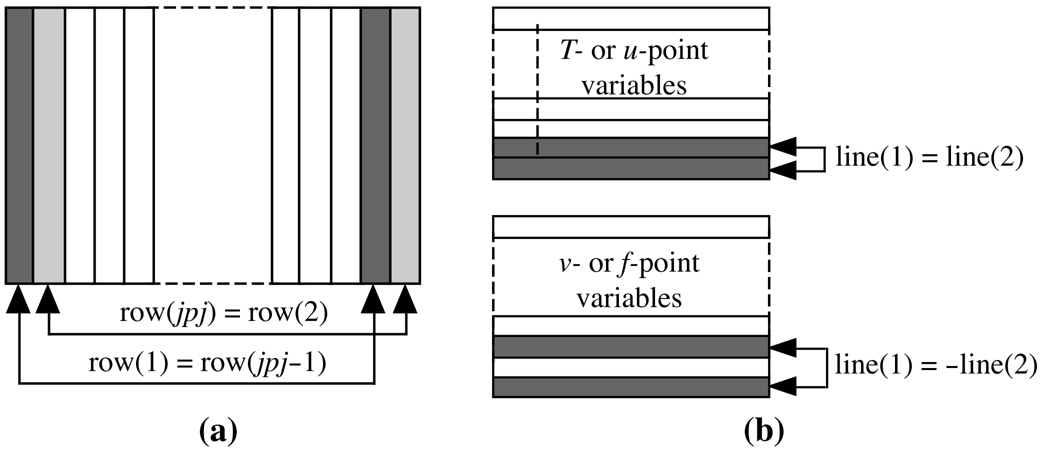


FIG. 7.3 – setting of (a) east-west cyclic (b) symmetric across the equator boundary conditions.

needed, it is set by a call to routine *lbclnk.F90*. The computation of momentum and tracer trends proceeds from $i = 2$ to $i = jpi - 1$ and from $j = 2$ to $j = jpj - 1$, $i.e.$ in the model interior. To choose a lateral model boundary condition is to specify the first and last rows and columns of the model variables.

For closed boundary (*jperio=0*) , solid walls are imposed at all model boundaries : first and last rows and columns are set to zero.

For cyclic east-west boundary (*jperio=1*) , first and last rows are set to zero (closed) whilst the first column is set to the value of the last-but-one column and the last column to the value of the second one (Fig. 7.2.1-a). Whatever flows out of the eastern (western) end of the basin enters the western (eastern) end. Note that there is no option for north-south cyclic or for doubly cyclic cases.

For symmetric boundary condition across the equator (*jperio=2*) , last rows, and first and last columns are set to zero (closed). The row of symmetry is chosen to be the u- and $T-$points equator line ($j = 2$, i.e. at the southern end of the domain). For arrays defined at $u-$ or $T-$points, the first row is set to the value of the third row while for most of v- and f-point arrays (v, ζ, $j\psi$, but scalar arrays such as eddy coefficients) the first row is set to minus the value of the second row (Fig. 7.2.1-b). Note that this boundary condition is not yet available for the case of a massively parallel computer (**key_mpp** defined).

7.2.2 North-fold (*jperio = 3* to 6)

The north fold boundary condition has been introduced in order to handle the north boundary of a three-polar ORCA grid. Such a grid has two poles in the northern hemis-

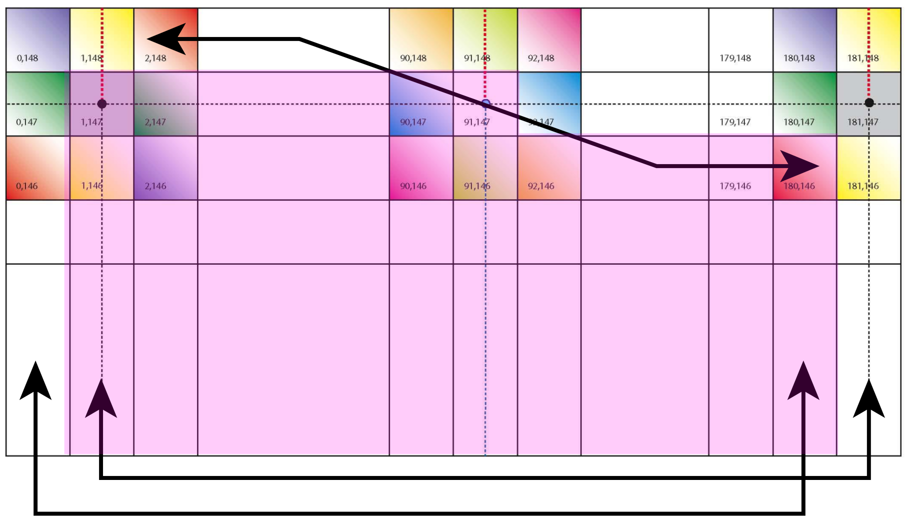


FIG. 7.4 – North fold boundary with a T-point pivot and cyclic east-west boundary condition ($jperio = 4$), as used in ORCA 2, 1/4, and 1/12. Pink shaded area corresponds to the inner domain mask (see text).

phere. to be completed...

7.3 Exchange with neighbouring processors (*lbclnk.F90*, *lib_mpp.F90*)

For massively parallel processing (mpp), a domain decomposition method is used. The basic idea of the method is to split the large computation domain of a numerical experiment into several smaller domains and solve the set of equations by addressing independent local problems. Each processor has its own local memory and computes the model equation over a subdomain of the whole model domain. The subdomain boundary conditions are specified through communications between processors which are organized by explicit statements (message passing method).

A big advantage is that the method does not need many modifications of the initial FORTRAN code. From the modeller's point of view, each sub domain running on a processor is identical to the "mono-domain" code. In addition, the programmer manages the communications between subdomains, and the code is faster when the number of processors is increased. The porting of OPA code on an iPSC860 was achieved during Guyon's PhD [Guyon et al. 1994, 1995] in collaboration with CETIIS and ONERA. The implementation in the operational context and the studies of performance on a T3D and T3E Cray computers have been made in collaboration with IDRIS and CNRS. The present implementation is largely inspired by Guyon's work [Guyon 1995].

The parallelization strategy is defined by the physical characteristics of the ocean model. Second order finite difference schemes lead to local discrete operators that depend at the very most on one neighbouring point. The only non-local computations concern the vertical physics (implicit diffusion, 1.5 turbulent closure scheme, ...) (delocalization over the whole water column), and the solving of the elliptic equation associated with the surface pressure gradient computation (delocalization over the whole horizontal domain). Therefore, a pencil strategy is used for the data sub-structuration : the 3D initial domain is laid out on local processor memories following a 2D horizontal topological splitting. Each sub-domain computes its own surface and bottom boundary conditions and has a side wall overlapping interface which defines the lateral boundary conditions for computations in the inner sub-domain. The overlapping area consists of the two rows at each edge of the sub-domain. After a computation, a communication phase starts : each processor sends to its neighbouring processors the update values of the points corresponding to the interior overlapping area to its neighbouring sub-domain (i.e. the innermost of the two overlapping rows). The communication is done through message passing. Usually the parallel virtual language, PVM, is used as it is a standard language available on nearly all MPP computers. More specific languages (i.e. computer dependant languages) can be easily used to speed up the communication, such as SHEM on a T3E computer. The data exchanges between processors are required at the very place where lateral domain boundary conditions are set in the mono-domain computation (§III.10-c) : the lbc_lnk routine which manages such conditions is substituted by mpplnk.F or mpplnk2.F routine when running on an MPP computer (**key_mpp_mpi** defined). It has to be pointed out that when using the MPP version of the model, the east-west cyclic boundary condition is done implicitly, whilst the south-symmetric boundary condition option is not available.

In the standard version of the OPA model, the splitting is regular and arithmetic. the i-axis is divided by *jpni* and the j-axis by *jpnj* for a number of processors *jpnij* most often equal to $jpni \times jpnj$ (model parameters set in *par_oce.F90*). Each processor is independent and without message passing or synchronous process , programs run alone and access just its own local memory. For this reason, the main model dimensions are now the local dimensions of the subdomain (pencil) that are named *jpi*, *jpj*, *jpk*. These dimensions include the internal domain and the overlapping rows. The number of rows to exchange (known as the halo) is usually set to one (*jpreci*=1, in *par_oce.F90*). The whole domain dimensions are named *jpiglo*, *jpjglo* and *jpk*. The relationship between the whole domain and a sub-domain is :

$$\begin{aligned} jpi &= (jpiglo - 2 * jpreci + (jpni - 1))/jpni + 2 * jpreci \\ jpj &= (jpjglo - 2 * jprecj + (jpnj - 1))/jpnj + 2 * jprecj \end{aligned} \tag{7.3}$$

where *jpni*, *jpnj* are the number of processors following the i- and j-axis.

Figure IV.3 : example of a domain splitting with 9 processors and no east-west cyclic boundary conditio

One also defines variables nldi and nlei which correspond to the internal domain bounds, and the variables nimpp and njmpp which are the position of the (1,1) grid-point in the global domain. An element of T_l, a local array (subdomain) corresponds to an ele-

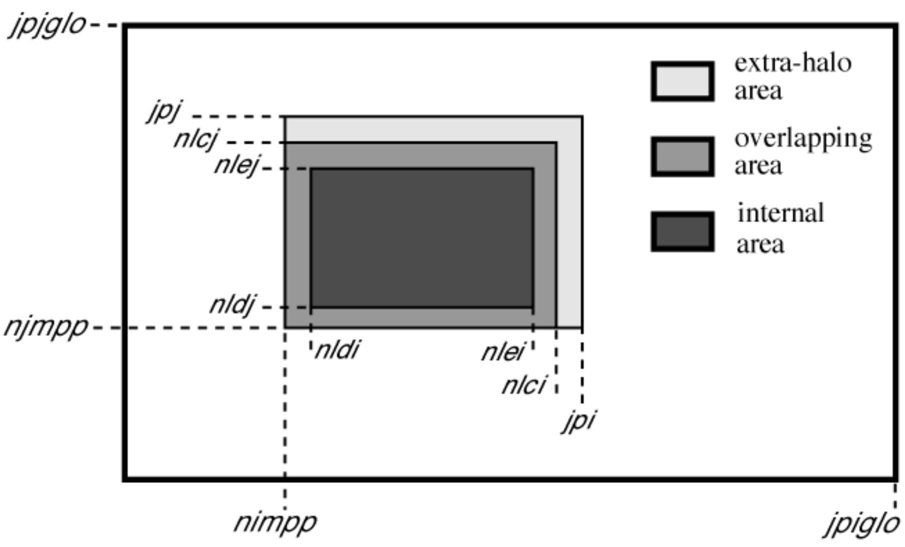


FIG. 7.5 – Positioning of a sub-domain when massively parallel processing is used.

ment of T_g, a global array (whole domain) by the relationship :

$$T_g(i + nimpp - 1, j + njmpp - 1, k) = T_l(i, j, k), \tag{7.4}$$

with $1 \leq i \leq jpi$, $1 \leq j \leq jpj$, and $1 \leq k \leq jpk$.

Processors are numbered from 0 to $jpnij - 1$, the number is saved in the variable nproc. In the standard version, a processor has no more than four neighbouring processors named nono (for north), noea (east), noso (south) and nowe (west) and two variables, nbondi and nbondj, indicate the relative position of the processor (see Fig.IV.3) :

– nbondi = -1 an east neighbour, no west processor,
– nbondi = 0 an east neighbour, a west neighbour,
– nbondi = 1 no east processor, a west neighbour,
– nbondi = 2 no splitting following the i-axis.

During the simulation, processors exchange data with their neighbours. If there is effectively a neighbour, the processor receives variables from this processor on its overlapping row, and sends the data issued from internal domain corresponding to the overlapping row of the other processor.

Figure IV.4 : pencil splitting with the additional outer halos

The *NEMO* model computes equation terms with the help of mask arrays (0 on land points and 1 on sea points). It is easily readable and very efficient in the context of a computer with vectorial architecture. However, in the case of a scalar processor, computations over the land regions become more expensive in terms of CPU time. It is worse when we use a complex configuration with a realistic bathymetry like the global ocean where more than 50 % of points are land points. For this reason, a pre-processing tool can be used to choose the mpp domain decomposition with a maximum number of only land points processors, which can then be eliminated. (For example, the mpp_optimiz tools, available from the DRAKKAR web site.) This optimisation is dependent on the specific bathymetry employed. The user then chooses optimal parameters *jpni*, *jpnj* and *jpnij* with $jpnij < jpni \times jpnj$, leading to the elimination of $jpni \times jpnj - jpnij$ land processors. When those parameters are specified in module *par_oce.F90*, the algorithm in the *inimpp2* routine sets each processor's parameters (nbound, nono, noea,...) so that the land-only processors are not taken into account.

Note that the inimpp2 routine is general so that the original inimpp routine should be suppressed from t

When land processors are eliminated, the value corresponding to these locations in the model output files is zero. Note that this is a problem for a mesh output file written by such a model configuration, because model users often divide by the scale factors ($e1t$, $e2t$, etc) and do not expect the grid size to be zero, even on land. It may be best not to eliminate land processors when running the model especially to write the mesh files as outputs (when *nmsh* namelist parameter differs from 0).

7.4 Open Boundary Conditions (key_obc)

```
!-----------------------------------------------------------------------
&namobc        !   open boundaries parameters                          ("key_obc")
!-----------------------------------------------------------------------
    nobc_dta    =    1      !  = 0 the obc data are equal to the initial state
                            !  = 1 the obc data are read in 'obc.dta' files
    cffile      = 'annual'  !  set to annual if obc datafile hold 1 year of data
                            !  set to monthly if obc datafile hold 1 month of data
    rdpein      =    1.     !  ???
    rdpwin      =    1.     !  ???
    rdpnin      =    1.     !  ???
    rdpsin      =    1.     !  ???
    rdpeob      = 3000.     !  time relaxation (days) for the east  open boundary
    rdpwob      =   15.     !    "        "             "     west         "
    rdpnob      = 3000.     !    "        "             "     north        "
    rdpsob      =   15.     !    "        "             "     south        "
    zbsic1      =  140.e+6  !   barotropic stream function on first  isolated coastline
    zbsic2      =    1.e+6  !    "                    "       second        "
    zbsic3      =    0.     !    "                    "       thrid         "
    ln_obc_clim= .false.    !  climatological obc data files (T) or not (F)
    ln_vol_cst = .true.     !  impose the total volume conservation (T) or not (F)
/
```

It is often necessary to implement a model configuration limited to an oceanic region or a basin, which communicates with the rest of the global ocean through ”open boundaries”. As stated by **?**, an open boundary is a computational border where the aim of the calculations is to allow the perturbations generated inside the computational domain to leave it without deterioration of the inner model solution. However, an open boundary also has to let information from the outer ocean enter the model and should support inflow and outflow conditions.

The open boundary package OBC is the first open boundary option developed in NEMO (originally in OPA8.2). It allows the user to

– tell the model that a boundary is ”open” and not closed by a wall, for example by modifying the calculation of the divergence of velocity there ;

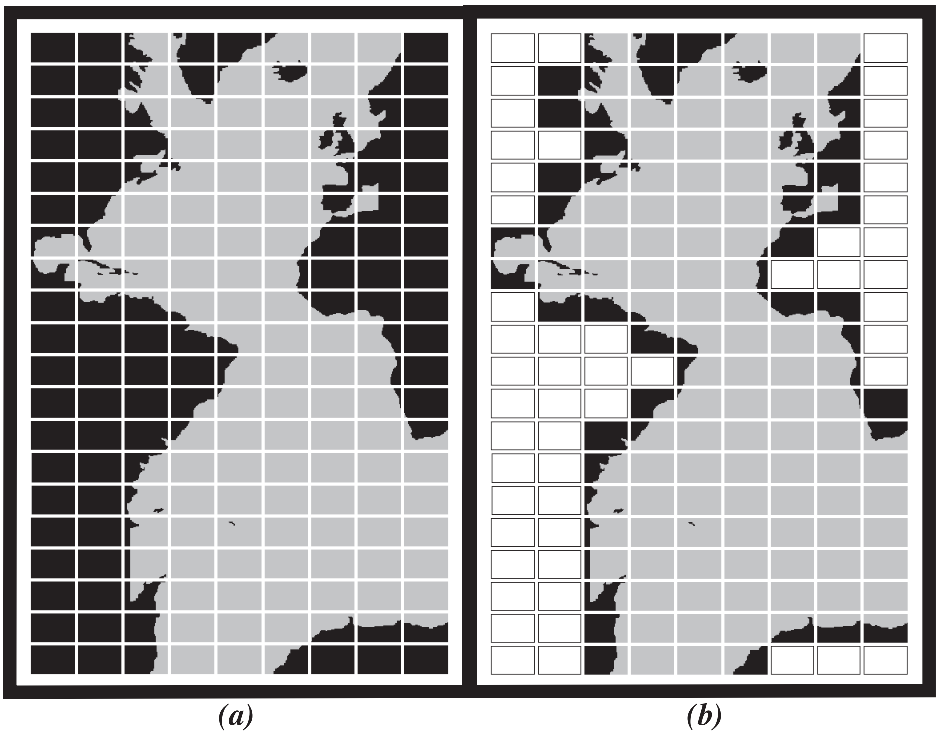


FIG. 7.6 – Example of Atlantic domain defined for the CLIPPER projet. Initial grid is composed of 773 x 1236 horizontal points. (a) the domain is split onto 9 subdomains (jpni=9, jpnj=20). 52 subdomains are land areas. (b) 52 subdomains are eliminated (white rectangles) and the resulting number of processors really used during the computation is jpnij=128.

– impose values of tracers and velocities at that boundary (values which may be taken from a climatology) : this is the“fixed OBC” option.
– calculate boundary values by a sophisticated algorithm combining radiation and relaxation (“radiative OBC” option)

The package resides in the OBC directory. It is described here in four parts : the boundary geometry (parameters to be set in *obc_par.F90*), the forcing data at the boundaries (module *obcdta.F90*), the radiation algorithm involving the namelist and module *obcrad.F90*, and a brief presentation of boundary update and restart files.

7.4.1 Boundary geometry

First one has to realize that open boundaries may not necessarily be located at the extremities of the computational domain. They may exist in the middle of the domain, for example at Gibraltar Straits if one wants to avoid including the Mediterranean in an

Atlantic domain. This flexibility has been found necessary for the CLIPPER project [**?**]. Because of the complexity of the geometry of ocean basins, it may even be necessary to have more than one ”west” open boundary, more than one ”north”, etc. This is not possible with the OBC option : only one open boundary of each kind, west, east, south and north is allowed ; these names refer to the grid geometry (not to the direction of the geographical ”west”, ”east”, etc).

The open boundary geometry is set by a series of parameters in the module *obc_par.F90*. For an eastern open boundary, parameters are *lp_obc_east* (true if an east open boundary exists), *jpieob* the i-index along which the eastern open boundary (eob) is located, *jpjed* the j-index at which it starts, and *jpjef* the j-index where it ends (note d is for ”début” and f for ”fin” in French). Similar parameters exist for the west, south and north cases (Table 7.4.1).

Boundary and Logical flag	Constant index	Starting index (début)	Ending index (fin)
West lp_obc_west	*jpiwob* $>= 2$ i-index of a u point	*jpjwd*$>= 2$ j of a T point	*jpjwf* ¡= *jpjglo*-1 j of a T point
East lp_obc_east	*jpieob*<=*jpiglo*-2 i-index of a u point	*jpjed* $>= 2$ j of a T point	*jpjef* <= *jpjglo*-1 j of a T point
South lp_obc_south	*jpjsob* $>= 2$ j-index of a v point	*jpisd* $>= 2$ i of a T point	*jpisf* <=*jpiglo*-1 i of a T point
North lp_obc_north	*jpjnob* <= *jpjglo*-2 j-index of a v point	*jpind* $>= 2$ i of a T point	*jpinf* <=*jpiglo*-1 i of a T point

TAB. 7.1 – Names of different indices relating to the open boundaries. In the case of a completely open ocean domain with four ocean boundaries, the parameters take exactly the values indicated.

The open boundaries must be along coordinate lines. On the C-grid, the boundary itself is along a line of normal velocity points : v points for a zonal open boundary (the south or north one), and u points for a meridional open boundary (the west or east one). Another constraint is that there still must be a row of masked points all around the domain, as if the domain were a closed basin (unless periodic conditions are used together with open boundary conditions). Therefore, an open boundary cannot be located at the first/last index, namely, 1, *jpiglo* or *jpjglo*. Also, the open boundary algorithm involves calculating the normal velocity points situated just on the boundary, as well as the tangential velocity and temperature and salinity just outside the boundary. This means that for a west/south boundary, normal velocities and temperature are calculated at the same index *jpiwob* and *jpjsob*, respectively. For an east/north boundary, the normal velocity is calculated at index *jpieob* and *jpjnob*, but the “outside” temperature is at index *jpieob*+1 and *jpjnob*+1. This means that *jpieob*, *jpjnob* cannot be bigger than *jpiglo*-2, *jpjglo*-2.

The starting and ending indices are to be thought of as T point indices : in many cases

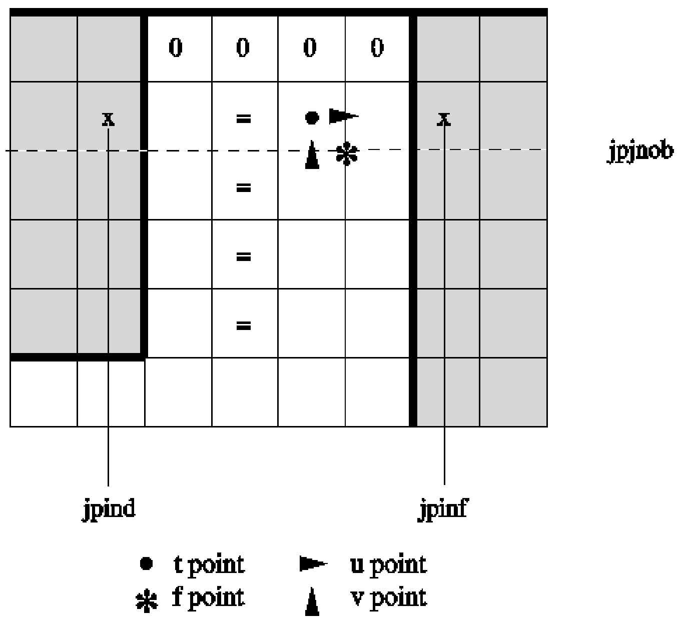


FIG. 7.7 – Localization of the North open boundary points.

they indicate the first land T-point, at the extremity of an open boundary (the coast line follows the f grid points, see Fig. 7.4.1 for an example of a northern open boundary). All indices are relative to the global domain. In the free surface case it is possible to have "ocean corners", that is, an open boundary starting and ending in the ocean.

Although not compulsory, it is highly recommended that the bathymetry in the vicinity of an open boundary follows the following rule : in the direction perpendicular to the open line, the water depth should be constant for 4 grid points. This is in order to ensure that the radiation condition, which involves model variables next to the boundary, is calculated in a consistent way. On Fig.7.4.1 we indicate by an $=$ symbol, the points which should have the same depth. It means that at the 4 points near the boundary, the bathymetry is cylindrical . The line behind the open T-line must be 0 in the bathymetry file (as shown on Fig.7.4.1 for example).

7.4.2 Boundary data

It is necessary to provide information at the boundaries. The simplest case is when this information does not change in time and is equal to the initial conditions (namelist variable *nobc_dta*=0). This is the case for the standard configuration EEL5 with open boundaries. When (*nobc_dta*=1), open boundary information is read from netcdf files. For convenience the input files are supposed to be similar to the "history" NEMO output files, for dimension names and variable names. Open boundary arrays must be dimensioned according to the parameters of table 7.4.1 : for example, at the western boundary, arrays have a dimension of *jpwf*-*jpwd*+1 in the horizontal and *jpk* in the vertical.

When ocean observations are used to generate the boundary data (a hydrographic section for example, as in **?**) it happens often that only the velocity normal to the boundary is known, which is the reason why the initial OBC code assumes that only T, S, and the normal velocity (u or v) needs to be specified. As more and more global model solutions and ocean analysis products become available, it will be possible to provide information about all the variables (including the tangential velocity) so that the specification of four variables at each boundaries will become standard. For the sea surface height, one must distinguish between the filtered free surface case and the time-splitting or explicit treatment of the free surface. In the first case, it is assumed that the user does not wish to represent high frequency motions such as tides. The boundary condition is thus one of zero normal gradient of sea surface height at the open boundaries, following **?**. No information other than the total velocity needs to be provided at the open boundaries in that case. In the other two cases (time splitting or explicit free surface), the user must provide barotropic information (sea surface height and barotropic velocities) and the use of the Flather algorithm for barotropic variables is recommanded. However, this algorithm has not yet been fully tested and bugs remain in NEMO v2.3. Users should read the code carefully before using it. Finally, in the case of the rigid lid approximation the barotropic streamfunction must be provided, as documented in **?**). This option is no longer recommended but remains in NEMO V2.3.

One frequently encountered case is when an open boundary domain is constructed from a global or larger scale NEMO configuration. Assuming the domain corresponds to indices $ib : ie$, $jb : je$ of the global domain, the bathymetry and forcing of the small domain can be created by using the following netcdf utility on the global files : ncks -F $-d\ x, ib, ie\ -d\ y, jb, je$ (part of the nco series of utilities, see http ://nco.sourceforge.net). The open boundary files can be constructed using ncks commands, following table 7.4.2.

It is assumed that the open boundary files contain the variables for the period of the model integration. If the boundary files contain one time frame, the boundary data is held fixed in time. If the files contain 12 values, it is assumed that the input is a climatology for a repeated annual cycle (corresponding to the case *ln_obc_clim* = .True.). The case of an arbitrary number of time frames is not yet implemented correctly ; the user is required to write his own code in the module *obc_dta.F90* to deal with this situation.

7.4.3 Radiation algorithm

The art of open boundary management consists in applying a constraint strong enough that the inner domain ”feels” the rest of the ocean, but weak enough that perturbations are allowed to leave the domain with minimum false reflections of energy. The constraints are specified separately at each boundary as time scales for ”inflow” and ”outflow” as defined below. The time scales are set (in days) by namelist parameters such as *rdpein*, *rdpeob* for the eastern open boundary for example. When both time scales are zero for a given boundary ($e.g.$ for the western boundary, *lp_obc_west*=.True., *rdpwob*=0 and *rdpwin*=0)

OBC	Variable	file name	Index	Start	end
West	T,S	obcwest_TS.nc	ib+1	jb+1	$je-1$
	U	obcwest_U.nc	ib+1	jb+1	$je-1$
	V	obcwest_V.nc	ib+1	jb+1	$je-1$
East	T,S	obceast_TS.nc	ie-1	jb+1	$je-1$
	U	obceast_U.nc	ie-2	jb+1	$je-1$
	V	obceast_V.nc	ie-1	jb+1	$je-1$
South	T,S	obcsouth_TS.nc	jb+1	ib+1	$ie-1$
	U	obcsouth_U.nc	jb+1	ib+1	$ie-1$
	V	obcsouth_V.nc	jb+1	ib+1	$ie-1$
North	T,S	obcnorth_TS.nc	je-1	ib+1	$ie-1$
	U	obcnorth_U.nc	je-1	ib+1	$ie-1$
	V	obcnorth_V.nc	je-2	ib+1	$ie-1$

TAB. 7.2 – Requirements for creating open boundary files from a global configuration, appropriate for the subdomain of indices $ib : ie$, $jb : je$. "Index" designates the i or j index along which the u of v boundary point is situated in the global configuration, starting and ending with the j or i indices indicated. For example, to generate file obcnorth_V.nc, use the command ncks $-F\ -d\ y, je-2$ $-d\ x, ib+1, ie-1$

this means that the boundary in question is a ”fixed ” boundary where the solution is set exactly by the boundary data. This is not recommended, except in combination with increased viscosity in a ”sponge” layer next to the boundary in order to avoid spurious reflections.

The radiationrelaxation algorithm is applied when either relaxation time (for ”inflow” or ”outflow”) is non-zero. It has been developed and tested in the SPEM model and its successor ROMS [**??**], which is an s-coordinate model on an Arakawa C-grid. Although the algorithm has been numerically successful in the CLIPPER Atlantic models, the physics do not work as expected [**?**]. Users are invited to consider open boundary conditions (OBC hereafter) with some scepticism [**??**].

The first part of the algorithm calculates a phase velocity to determine whether perturbations tend to propagate toward, or away from, the boundary. Let us consider a model variable ϕ. The phase velocities ($C_{\phi x}$,$C_{\phi y}$) for the variable ϕ, in the directions normal and tangential to the boundary are

$$C_{\phi x} = \frac{-\phi_t}{(\phi_x^2 + \phi_y^2)}\phi_x \qquad C_{\phi y} = \frac{-\phi_t}{(\phi_x^2 + \phi_y^2)}\phi_y. \tag{7.5}$$

Following **?** and **?** we retain only the normal component of the velocity, $C_{\phi x}$, setting $C_{\phi y} = 0$ (but unlike the original Orlanski radiation algorithm we retain ϕ_y in the expression for $C_{\phi x}$).

The discrete form of (7.5), described by **?**, takes into account the two rows of grid points situated inside the domain next to the boundary, and the three previous time steps (n, $n-1$, and $n-2$). The same equation can then be discretized at the boundary at time steps $n-1$, n and $n+1$ in order to extrapolate for the new boundary value ϕ^{n+1}.

In the open boundary algorithm as implemented in NEMO v2.3, the new boundary values are updated differently depending on the sign of $C_{\phi x}$. Let us take an eastern boundary as an example. The solution for variable ϕ at the boundary is given by a generalized wave equation with phase velocity C_ϕ, with the addition of a relaxation term, as :

$$\phi_t = -C_{\phi x}\phi_x + \frac{1}{\tau_o}(\phi_c - \phi) \qquad (C_{\phi x} > 0), \tag{7.6}$$

$$\phi_t = \frac{1}{\tau_i}(\phi_c - \phi) \qquad (C_{\phi x} < 0), \tag{7.7}$$

where ϕ_c is the estimate of ϕ at the boundary, provided as boundary data. Note that in (7.6), $C_{\phi x}$ is bounded by the ratio $\delta x/\delta t$ for stability reasons. When $C_{\phi x}$ is eastward (outward propagation), the radiation condition (7.6) is used. When $C_{\phi x}$ is westward (inward propagation), (7.7) is used with a strong relaxation to climatology (usually $\tau_i =$ *rdpein* $=$1 day). Equation (7.7) is solved with a Euler time-stepping scheme. As a consequence, setting τ_i smaller than, or equal to the time step is equivalent to a fixed boundary condition. A time scale of one day is usually a good compromise which guarantees that the inflow conditions remain close to climatology while ensuring numerical stability.

In the case of a western boundary located in the Eastern Atlantic, **?** have been able to implement the radiation algorithm without any boundary data, using persistence from the previous time step instead. This solution has not worked in other cases [**?**], so that the use of boundary data is recommended. Even in the outflow condition (7.6), we have found it desirable to maintain a weak relaxation to climatology. The time step is usually chosen so as to be larger than typical turbulent scales (of order 1000 days).

The radiation condition is applied to the model variables : temperature, salinity, tangential and normal velocities. For normal and tangential velocities, u and v, radiation is applied with phase velocities calculated from u and v respectively. For the radiation of tracers, we use the phase velocity calculated from the tangential velocity in order to avoid calculating too many independent radiation velocities and because tangential velocities and tracers have the same position along the boundary on a C-grid.

7.4.4 Domain decomposition (key_mpp_mpi)

When **key_mpp_mpi** is active in the code, the computational domain is divided into rectangles that are attributed each to a different processor. The open boundary code is "mpp-compatible" up to a certain point. The radiation algorithm will not work if there is an mpp subdomain boundary parallel to the open boundary at the index of the boundary, or the grid point after (outside), or three grid points before (inside). On the other hand, there is no problem if an mpp subdomain boundary cuts the open boundary perpendicularly. These geometrical limitations must be checked for by the user (there is no safeguard in the code). The general principle for the open boundary mpp code is that loops over the open boundaries not sure what this means are performed on local indices (nie0, nie1, nje0, nje1 for an eastern boundary for instance) that are initialized in module *obc_ini.F90*. Those indices have relevant values on the processors that contain a segment of an open boundary. For processors that do not include an open boundary segment, the indices are such that the calculations within the loops are not performed.

Arrays of climatological data that are read from files are seen by all processors and have the same dimensions for all (for instance, for the eastern boundary, uedta(jpjglo,jpk,2)). On the other hand, the arrays for the calculation of radiation are local to each processor (uebnd(jpj,jpk,3,3) for instance). This allowed the CLIPPER model for example, to save on memory where the eastern boundary crossed 8 processors so that *jpj* was much smaller than (*jpjef*-*jpjed*+1).

7.4.5 Volume conservation

It is necessary to control the volume inside a domain when using open boundaries. With fixed boundaries, it is enough to ensure that the total inflow/outflow has reasonable values (either zero or a value compatible with an observed volume balance). When using radiative boundary conditions it is necessary to have a volume constraint because each open boundary works independently from the others. The methodology used to control this volume is identical to the one coded in the ROMS model [**?**].

Explain obc_vol…

OBC algorithm for update, OBC restart, list of routines where obc key appears…

OBC rigid lid ? . . .

7.5 Flow Relaxation Scheme (? ? ?)

Lateral Ocean Physics (LDF)

Contents

The lateral physics terms in the momentum and tracer equations have been described in §2.6.1 and their discrete formulation in §4.2 and §5.5). In this section we further discuss each lateral physics option. Choosing one lateral physics scheme means for the user defining, (1) the space and time variations of the eddy coefficients ; (2) the direction along which the lateral diffusive fluxes are evaluated (model level, geopotential or isopycnal surfaces) ; and (3) the type of operator used (harmonic, or biharmonic operators, and for tracers only, eddy induced advection on tracers). These three aspects of the lateral diffusion are set through namelist parameters and CPP keys (see the nam_traldf and nam_dynldf below).

```
!----------------------------------------------------------------------
&nam_traldf    !   lateral diffusion scheme for tracer
!----------------------------------------------------------------------
!                               !  Type of the operator :
   ln_traldf_lap    =  .true.   !     laplacian operator
   ln_traldf_bilap  =  .false.  !     bilaplacian operator
                                !  Direction of action  :
   ln_traldf_level  =  .false.  !     iso-level
   ln_traldf_hor    =  .false.  !     horizontal (geopotential)         (require "key_ldfslp" when ln_sco=T)
   ln_traldf_iso    =  .true.   !     iso-neutral                       (require "key_ldfslp")
!                               !  Coefficient
```

```
   aht0        =  2000.        !     horizontal eddy diffusivity for tracers [m2/s]
   ahtb0       =     0.        !     background eddy diffusivity for ldf_iso [m2/s]
   aeiv0       =  2000.        !     eddy induced velocity coefficient [m2/s]    (require "key_traldf_eiv")
/


!-----------------------------------------------------------------------
&nam_dynldf    !   lateral diffusion on momentum
!-----------------------------------------------------------------------
!                              !  Type of the operator :
   ln_dynldf_lap    =  .true.   !     laplacian operator
   ln_dynldf_bilap  =  .false.  !     bilaplacian operator
!                              !  Direction of action  :
   ln_dynldf_level  =  .false.  !     iso-level
   ln_dynldf_hor    =  .true.   !     horizontal (geopotential)        (require "key_ldfslp" in s-coord.)
   ln_dynldf_iso    =  .false.  !     iso-neutral                      (require "key_ldfslp")
                               !  Coefficient
   rn_ahm_0_lap     = 40000.    !     horizontal laplacian eddy viscosity   [m2/s]
   rn_ahmb_0        =     0.    !     background eddy viscosity for ldf_iso [m2/s]
   rn_ahm_0_blp     =     0.    !     horizontal bilaplacian eddy viscosity [m4/s]
/
```

8.1 Lateral Mixing Coefficient (*ldftra.F90, ldfdyn).F90*

Introducing a space variation in the lateral eddy mixing coefficients changes the model core memory requirement, adding up to four extra three-dimensional arrays for the geopotential or isopycnal second order operator applied to momentum. Six CPP keys control the space variation of eddy coefficients : three for momentum and three for tracer. The three choices allow : a space variation in the three space directions, in the horizontal plane, or in the vertical only. The default option is a constant value over the whole ocean on both momentum and tracers.

The number of additional arrays that have to be defined and the gridpoint position at which they are defined depend on both the space variation chosen and the type of operator used. The resulting eddy viscosity and diffusivity coefficients can be a function of more than one variable. Changes in the computer code when switching from one option to another have been minimized by introducing the eddy coefficients as statement functions (include file *ldftra_substitute.h90* and *ldfdyn_substitute.h90*). The functions are replaced by their actual meaning during the preprocessing step (CPP). The specification of the space variation of the coefficient is made in *ldftra.F90* and *ldfdyn.F90*, or more precisely in include files *ldftra_cNd.h90* and *ldfdyn_cNd.h90*, with N=1, 2 or 3. The user can modify these include files as he/she wishes. The way the mixing coefficient are set in the reference version can be briefly described as follows :

Constant Mixing Coefficients (default option)

When none of the **key_ldfdyn_...** and **key_ldftra_...** keys are defined, a constant value is used over the whole ocean for momentum and tracers, which is specified through the *ahm0* and *aht0* namelist parameters.

Vertically varying Mixing Coefficients (key_ldftra_c1d and key_ldfdyn_c1d)

The 1D option is only available when using the z-coordinate with full step. Indeed in all the other types of vertical coordinate, the depth is a 3D function of (**i**,**j**,**k**) and therefore, introducing depth-dependent mixing coefficients will require 3D arrays, $i.e.$ **key_ldftra_c3d** and **key_ldftra_c3d**. In the 1D option, a hyperbolic variation of the lateral mixing coefficient is introduced in which the surface value is *aht0* (*ahm0*), the bottom value is 1/4 of the surface value, and the transition takes place around z=300 m with a width of 300 m ($i.e.$ both the depth and the width of the inflection point are set to 300 m). This profile is hard coded in file *ldftra_c1d.h90*, but can be easily modified by users.

Horizontally Varying Mixing Coefficients (key_ldftra_c2d and key_ldfdyn_c2d)

By default the horizontal variation of the eddy coefficient depends on the local mesh size and the type of operator used :

$$A_l = \left\{ \begin{aligned} & \frac{\max(e_1,e_2)}{e_{max}} A_o^l & \text{for laplacian operator} \\ & \frac{\max(e_1,e_2)^3}{e_{max}^3} A_o^l & \text{for bilaplacian operator} \end{aligned} \right. \qquad \text{comments} \tag{8.1}$$

where e_{max} is the maximum of e_1 and e_2 taken over the whole masked ocean domain, and A_o^l is the *ahm0* (momentum) or *aht0* (tracer) namelist parameter. This variation is intended to reflect the lesser need for subgrid scale eddy mixing where the grid size is smaller in the domain. It was introduced in the context of the DYNAMO modelling project [**?**].

Other formulations can be introduced by the user for a given configuration. For example, in the ORCA2 global ocean model (**key_orca_r2**), the laplacian viscosity operator uses *ahm0* = $4.10^4 m^2/s$ poleward of 20° north and south and decreases linearly to *aht0* = $2.10^3 m^2/s$ at the equator [**??**]. This modification can be found in routine *ldf_dyn_c2d_orca* defined in *ldfdyn_c2d.F90*. Similar modified horizontal variations can be found with the Antarctic or Arctic sub-domain options of ORCA2 and ORCA05 (**key_antarctic** or **key_arctic** defined, see *ldfdyn_antarctic.h90* and *ldfdyn_arctic.h90*).

Space Varying Mixing Coefficients (key_ldftra_c3d and key_ldfdyn_c3d)

The 3D space variation of the mixing coefficient is simply the combination of the 1D and 2D cases, $i.e.$ a hyperbolic tangent variation with depth associated with a grid size dependence of the magnitude of the coefficient.

Space and Time Varying Mixing Coefficients

There is no default specification of space and time varying mixing coefficient. The only case available is specific to the ORCA2 and ORCA05 global ocean configurations (**key_orca_r2** or **key_orca_r05**). It provides only a tracer mixing coefficient for eddy induced velocity (ORCA2) or both iso-neutral and eddy induced velocity (ORCA05) that depends on the local growth rate of baroclinic instability. This specification is actually used when an ORCA key and both **key_traldf_eiv** and **key_traldf_c2d** are defined.

A space variation in the eddy coefficient appeals several remarks :

(1) the momentum diffusion operator acting along model level surfaces is written in terms of curl and divergent components of the horizontal current (see §2.6.2). Although the eddy coefficient can be set to different values in these two terms, this option is not available.

(2) with an horizontally varying viscosity, the quadratic integral constraints on enstrophy and on the square of the horizontal divergence for operators acting along model-surfaces are no longer satisfied (Appendix C.3).

(3) for isopycnal diffusion on momentum or tracers, an additional purely horizontal background diffusion with uniform coefficient can be added by setting a non zero value of *ahmb0* or *ahtb0*, a background horizontal eddy viscosity or diffusivity coefficient (namelist parameters whose default values are 0). However, the technique used to compute the isopycnal slopes is intended to get rid of such a background diffusion, since it introduces spurious diapycnal diffusion (see §8.2).

(4) when an eddy induced advection term is used (**key_trahdf_eiv**), A^{eiv}, the eddy induced coefficient has to be defined. Its space variations are controlled by the same CPP variable as for the eddy diffusivity coefficient ($i.e.$ **key_traldf_cNd**).

(5) the eddy coefficient associated with a biharmonic operator must be set to a *negative* value.

(6) it is possible to use both the laplacian and biharmonic operators concurrently.

(7) for testing purposes it is possible to run without lateral diffusion on momentum.

8.2 Direction of Lateral Mixing (*ldfslp.F90*)

A direction for lateral mixing has to be defined when the desired operator does not act along the model levels. This occurs when (a) horizontal mixing is required on tracer or momentum (*ln_traldf_hor* or *ln_dynldf_hor*) in s- or mixed s-z- coordinates, and (b) isoneutral mixing is required whatever the vertical coordinate is. This direction of mixing is defined by its slopes in the **i-** and **j-**directions at the face of the cell of the quantity to be diffused. For a tracer, this leads to the following four slopes : r_{1u}, r_{1w}, r_{2v}, r_{2w} (see (4.11)), while for momentum the slopes are r_{1T}, r_{1uw}, r_{2f}, r_{2uw} for u and r_{1f}, r_{1vw}, r_{2T}, r_{2vw} for v.

8.2.1 slopes for tracer geopotential mixing in the s-coordinate

In s-coordinates, geopotential mixing ($i.e.$ horizontal mixing) r_1 and r_2 are the slopes between the geopotential and computational surfaces. Their discrete formulation is found by locally solving (4.11) when the diffusive fluxes in the three directions are set to zero and T is assumed to be horizontally uniform, $i.e.$ a linear function of z_T, the depth of a T-point.

$$
\begin{aligned}
r_{1u} &= \frac{e_{3u}}{\left(e_{1u}\ \overline{\overline{e_{3w}}}^{\,i+1/2,\,k}\right)}\ \delta_{i+1/2}[z_T] &&\approx \frac{1}{e_{1u}}\ \delta_{i+1/2}[z_T] \\
r_{2v} &= \frac{e_{3v}}{\left(e_{2v}\ \overline{\overline{e_{3w}}}^{\,j+1/2,\,k}\right)}\ \delta_{j+1/2}[z_T] &&\approx \frac{1}{e_{2v}}\ \delta_{j+1/2}[z_T] \\
r_{1w} &= \frac{1}{e_{1w}}\ \overline{\overline{\delta_{i+1/2}[z_T]}}^{\,i,\,k+1/2} &&\approx \frac{1}{e_{1w}}\ \delta_{i+1/2}[z_{uw}] \\
r_{2w} &= \frac{1}{e_{2w}}\ \overline{\overline{\delta_{j+1/2}[z_T]}}^{\,j,\,k+1/2} &&\approx \frac{1}{e_{2w}}\ \delta_{j+1/2}[z_{vw}]
\end{aligned}
\tag{8.2}
$$

These slopes are computed once in *ldfslp_init* when *ln_sco*=True, and either *ln_traldf_hor*=True or *ln_dynldf_hor*=True.

8.2.2 slopes for tracer iso-neutral mixing

In iso-neutral mixing r_1 and r_2 are the slopes between the iso-neutral and computational surfaces. Their formulation does not depend on the vertical coordinate used. Their discrete formulation is found using the fact that the diffusive fluxes of locally referenced potential density ($i.e.$ $insitu$ density) vanish. So, substituting T by ρ in (4.11) and setting the diffusive fluxes in the three directions to zero leads to the following definition for the neutral slopes :

$$
\begin{aligned}
r_{1u} &= \frac{e_{3u}}{e_{1u}}\ \frac{\delta_{i+1/2}[\rho]}{\overline{\overline{\delta_{k+1/2}[\rho]}}^{\,i+1/2,\,k}} \\
r_{2v} &= \frac{e_{3v}}{e_{2v}}\ \frac{\delta_{j+1/2}\,[\rho]}{\overline{\overline{\delta_{k+1/2}[\rho]}}^{\,j+1/2,\,k}} \\
r_{1w} &= \frac{e_{3w}}{e_{1w}}\ \frac{\overline{\overline{\delta_{i+1/2}[\rho]}}^{\,i,\,k+1/2}}{\delta_{k+1/2}[\rho]} \\
r_{2w} &= \frac{e_{3w}}{e_{2w}}\ \frac{\overline{\overline{\delta_{j+1/2}[\rho]}}^{\,j,\,k+1/2}}{\delta_{k+1/2}[\rho]}
\end{aligned}
\tag{8.3}
$$

As the mixing is performed along neutral surfaces, the gradient of ρ in (8.3) has to be evaluated at the same local pressure (which, in decibars, is approximated by the depth in meters in the model). Therefore (8.3) cannot be used as such, but further transformation is needed depending on the vertical coordinate used :

z-coordinate with full step : in (8.3) the densities appearing in the i and j derivatives are taken at the same depth, thus the $insitu$ density can be used. This is not the case for the vertical derivatives : $\delta_{k+1/2}[\rho]$ is replaced by $-\rho N^2/g$, where N^2 is the local Brunt-Vaisälä frequency evaluated following **?** (see §4.8.2).

z-coordinate with partial step : this case is identical to the full step case except that at partial step level, the *horizontal* density gradient is evaluated as described in §4.9.

s- or hybrid s-z- coordinate : in the current release of *NEMO* , there is no specific treatment for iso-neutral mixing in the s-coordinate. In other words, iso-neutral mixing will only be accurately represented with a linear equation of state (*neos*=1 or 2). In the case of a ”true” equation of state, the evaluation of i and j derivatives in (8.3) will include a pressure dependent part, leading to the wrong evaluation of the neutral slopes.

Note : The solution for s-coordinate passes trough the use of different (and better) expression for the constraint on iso-neutral fluxes. Following **?**, instead of specifying directly that there is a zero neutral diffusive flux of locally referenced potential density, we stay in the T-S plane and consider the balance between the neutral direction diffusive fluxes of potential temperature and salinity :

$$\alpha\,\mathbf{F}(T) = \beta\,\mathbf{F}(S) \tag{8.4}$$

This constraint leads to the following definition for the slopes :

$$\begin{aligned}
r_{1u} &= \frac{e_{3u}}{e_{1u}} \; \frac{\alpha_u\;\delta_{i+1/2}[T] - \beta_u\;\delta_{i+1/2}[S]}{\alpha_u\;\overline{\overline{\delta_{k+1/2}[T]}}^{\,i+1/2,\,k} - \beta_u\;\overline{\overline{\delta_{k+1/2}[S]}}^{\,i+1/2,\,k}} \\
r_{2v} &= \frac{e_{3v}}{e_{2v}} \; \frac{\alpha_v\;\delta_{j+1/2}[T] - \beta_v\;\delta_{j+1/2}[S]}{\alpha_v\;\overline{\overline{\delta_{k+1/2}[T]}}^{\,j+1/2,\,k} - \beta_v\;\overline{\overline{\delta_{k+1/2}[S]}}^{\,j+1/2,\,k}} \\
r_{1w} &= \frac{e_{3w}}{e_{1w}} \; \frac{\alpha_w\;\overline{\overline{\delta_{i+1/2}[T]}}^{\,i,\,k+1/2} - \beta_w\;\overline{\overline{\delta_{i+1/2}[S]}}^{\,i,\,k+1/2}}{\alpha_w\;\delta_{k+1/2}[T] - \beta_w\;\delta_{k+1/2}[S]} \\
r_{2w} &= \frac{e_{3w}}{e_{2w}} \; \frac{\alpha_w\;\overline{\overline{\delta_{j+1/2}[T]}}^{\,j,\,k+1/2} - \beta_w\;\overline{\overline{\delta_{j+1/2}[S]}}^{\,j,\,k+1/2}}{\alpha_w\;\delta_{k+1/2}[T] - \beta_w\;\delta_{k+1/2}[S]}
\end{aligned} \tag{8.5}$$

where α and β, the thermal expansion and saline contraction coefficients introduced in §4.8.2, have to be evaluated at the three velocity points. In order to save computation time, they should be approximated by the mean of their values at T-points (for example in the case of α : $\alpha_u = \overline{\alpha_T}^{i+1/2}$, $\alpha_v = \overline{\alpha_T}^{j+1/2}$ and $\alpha_w = \overline{\alpha_T}^{k+1/2}$).

Note that such a formulation could be also used in the z-coordinate and z-coordinate with partial steps cases.

This implementation is a rather old one. It is similar to the one proposed by Cox [1987], except for the background horizontal diffusion. Indeed, the Cox implementation of isopycnal diffusion in GFDL-type models requires a minimum background horizontal diffusion for numerical stability reasons. To overcome this problem, several techniques have been proposed in which the numerical schemes of the ocean model are modified [**??**]. Here, another strategy has been chosen [**?**] : a local filtering of the iso-neutral slopes (made on 9 grid-points) prevents the development of grid point noise generated by the iso-neutral diffusion operator (Fig. 8.2.2). This allows an iso-neutral diffusion scheme without additional background horizontal mixing. This technique can be viewed as a diffusion operator that acts along large-scale (2 Δx) iso-neutral surfaces. The diapycnal diffusion required for numerical stability is thus minimized and its net effect on the flow is quite small when compared to the effect of an horizontal background mixing.

Nevertheless, this iso-neutral operator does not ensure that variance cannot increase, contrary to the **?** operator which has that property.

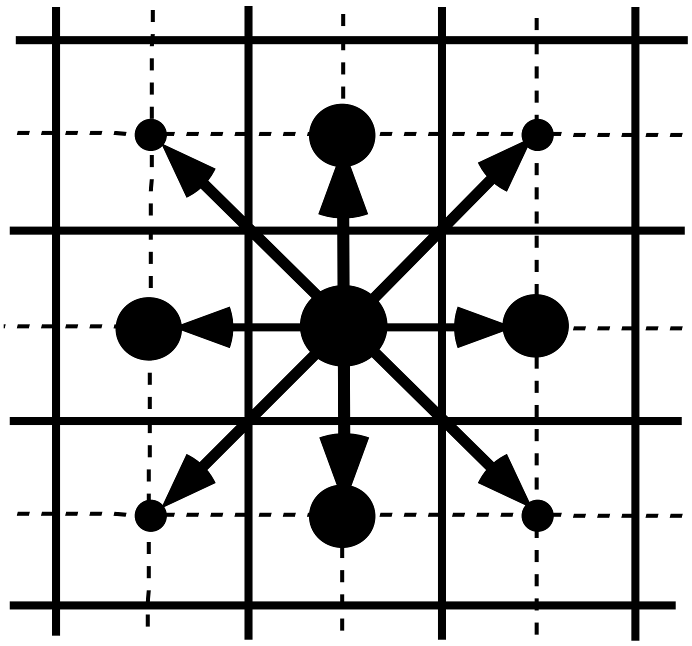


FIG. 8.1 – averaging procedure for isopycnal slope computation.

In addition and also for numerical stability reasons [**??**], the slopes are bounded by $1/100$ everywhere. This limit is decreasing linearly to zero fom 70 meters depth and the surface (the fact that the eddies "feel" the surface motivates this flattening of isopycnals near the surface).

For numerical stability reasons [**??**], the slopes must also be bounded by $1/100$ everywhere. This constraint is applied in a piecewise linear fashion, increasing from zero at the surface to $1/100$ at 70 metres and thereafter decreasing to zero at the bottom of the ocean. (the fact that the eddies "feel" the surface motivates this flattening of isopycnals near the surface).

add here a discussion about the flattening of the slopes, vs tapering the coefficient.

8.2.3 slopes for momentum iso-neutral mixing

The iso-neutral diffusion operator on momentum is the same as the one used on tracers but applied to each component of the velocity separately (see (5.25) in section 5.5.2). The slopes between the surface along which the diffusion operator acts and the surface of computation (z- or s-surfaces) are defined at T-, f-, and *uw*- points for the u-component, and T-, f- and *vw*- points for the v-component. They are computed from the slopes used for tracer diffusion, $i.e.$ (8.2) and (8.3) :

$$\begin{aligned} r_{1T} &= \overline{r_{1u}}^{\,i} & r_{1f} &= \overline{r_{1u}}^{\,i+1/2} \\ r_{2f} &= \overline{r_{2v}}^{\,j+1/2} & r_{2T} &= \overline{r_{2v}}^{\,j} \\ r_{1uw} &= \overline{r_{1w}}^{\,i+1/2} \quad \text{and} & r_{1vw} &= \overline{r_{1w}}^{\,j+1/2} \\ r_{2uw} &= \overline{r_{2w}}^{\,j+1/2} & r_{2vw} &= \overline{r_{2w}}^{\,j+1/2} \end{aligned} \tag{8.6}$$

The major issue remaining is in the specification of the boundary conditions. The same boundary conditions are chosen as those used for lateral diffusion along model level surfaces, i.e. using the shear computed along the model levels and with no additional friction at the ocean bottom (see §7.1).

8.3 Eddy Induced Velocity (*traadv_eiv.F90, ldfeiv.F90*)

When Gent and McWilliams [1990] diffusion is used (**key_traldf_eiv** defined), an eddy induced tracer advection term is added, the formulation of which depends on the slopes of iso-neutral surfaces. Contrary to the case of iso-neutral mixing, the slopes used here are referenced to the geopotential surfaces, $i.e.$ (8.2) is used in z-coordinates, and the sum (8.2) + (8.3) in s-coordinates. The eddy induced velocity is given by :

$$\begin{aligned} u^* &= \frac{1}{e_{2u}e_{3u}}\;\delta_k\left[e_{2u}\,A_{uw}^{eiv}\;\overline{r_{1w}}^{\,i+1/2}\right] \\ v^* &= \frac{1}{e_{1u}e_{3v}}\;\delta_k\left[e_{1v}\,A_{vw}^{eiv}\;\overline{r_{2w}}^{\,j+1/2}\right] \\ w^* &= \frac{1}{e_{1w}e_{2w}}\;\left\{\delta_i\left[e_{2u}\,A_{uw}^{eiv}\;\overline{r_{1w}}^{\,i+1/2}\right]+\delta_j\left[e_{1v}\,A_{vw}^{eiv}\;\overline{r_{2w}}^{\,j+1/2}\right]\right\} \end{aligned} \tag{8.7}$$

where A^{eiv} is the eddy induced velocity coefficient whose value is set through *aeiv*, a *nam_traldf* namelist parameter. The three components of the eddy induced velocity are computed and add to the eulerian velocity in *traadv_eiv.F90*. This has been preferred to a separate computation of the advective trends associated with the eiv velocity, since it allows us to take advantage of all the advection schemes offered for the tracers (see §4.1) and not just the 2^{nd} order advection scheme as in previous releases of OPA [**?**]. This is particularly useful for passive tracers where *positivity* of the advection scheme is of paramount importance.

At the surface, lateral and bottom boundaries, the eddy induced velocity, and thus the advective eddy fluxes of heat and salt, are set to zero.

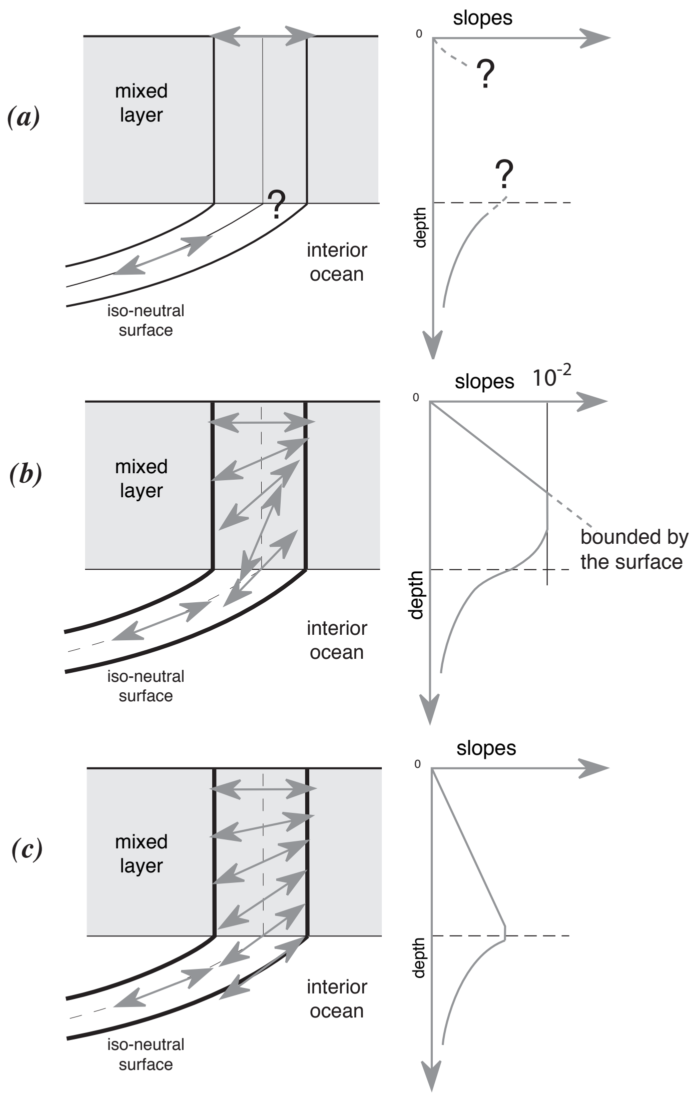


FIG. 8.2 – Vertical profile of the slope used for lateral mixing in the mixed layer : *(a)* in the real ocean the slope is the iso-neutral slope in the ocean interior, which has to be adjusted at the surface boundary (i.e. it must tend to zero at the surface since there is no mixing across the air-sea interface : wall boundary condition). Nevertheless, the profile between the surface zero value and the interior iso-neutral one is unknown, and especially the value at the base of the mixed layer ; *(b)* profile of slope using a linear tapering of the slope near the surface and imposing a maximum slope of 1/100 ; *(c)* profile of slope actually used in *NEMO* : a linear decrease of the slope from zero at the surface to its ocean interior value computed just below the mixed layer. Note the huge change in the slope at the base of the mixed layer between *(b)* and *(c)*.

Vertical Ocean Physics (ZDF)

Contents

9.1 Vertical Mixing

The discrete form of the ocean subgrid scale physics has been presented in §4.3 and §5.6. At the surface and bottom boundaries, the turbulent fluxes of momentum, heat and salt have to be defined. At the surface they are prescribed from the surface forcing (see Chap. 6), while at the bottom they are set to zero for heat and salt, unless a geothermal flux forcing is prescribed as a bottom boundary condition ($i.e.$ **key_trabbl** defined, see §4.4.3), and specified through a bottom friction parameterisation for momentum (see §9.4).

In this section we briefly discuss the various choices offered to compute the vertical eddy viscosity and diffusivity coefficients, A_u^{vm} , A_v^{vm} and A^{vT} (A^{vS}), defined at uw-, vw- and w- points, respectively (see §4.3 and §5.6). These coefficients can be assumed to be either constant, or a function of the local Richardson number, or computed from a turbulent closure model (either TKE or KPP formulation). The computation of these coefficients is initialized in the *zdfini.F90* module and performed in the *zdfric.F90, zdftke.F90* or *zdfkpp.F90* modules. The trends due to the vertical momentum and tracer diffusion, including the surface forcing, are computed and added to the general trend in the *dynzdf.F90* and *trazdf.F90* modules, respectively. These trends can be computed using either a forward time stepping scheme (namelist parameter *np_zdfexp*=true) or a backward time stepping scheme (*np_zdfexp*=false) depending on the magnitude of the mixing coefficients, and thus of the formulation used (see §3.4).

9.1.1 Constant (key_zdfcst)

```
!-----------------------------------------------------------------------
&namzdf        !   vertical physics
!-----------------------------------------------------------------------
   avm0        =   1.2e-4  !  vertical eddy viscosity   [m2/s]          (background Kz if not "key_zdfcst")
   avt0        =   1.2e-5  !  vertical eddy diffusivity [m2/s]          (background Kz if not "key_zdfcst")
   ln_zdfnpc   = .false.   !  convection: Non-Penetrative algorithm (T) or not (F)
   ln_zdfevd   = .true.    !  convection: enhanced vertical diffusion (T) or not (F)
   avevd       = 100.      !  vertical coefficient for enhanced diffusion scheme [m2/s]
   n_evdm      =   1       !  enhanced mixing apply on tracer (=0) or on tracer and momentum (=1)
   ln_zdfexp   =  .false.  !  split explicit (T) or implicit (F) time stepping
   n_zdfexp    =   3       !  number of sub-timestep for ln_zdfexp=T
/
```

When **key_zdfcst** is defined, the momentum and tracer vertical eddy coefficients are set to constant values over the whole ocean. This is the crudest way to define the vertical ocean physics. It is recommended that this option is only used in process studies, not in basin scale simulations. Typical values used in this case are :

$$A_u^{vm} = A_v^{vm} = 1.2\ 10^{-4}\ m^2.s^{-1}$$

$$A^{vT} = A^{vS} = 1.2\ 10^{-5}\ m^2.s^{-1}$$

These values are set through the *avm0* and *avt0* namelist parameters. In all cases, do not use values smaller that those associated with the molecular viscosity and diffusivity, that is $\sim\ 10^{-6}\ m^2.s^{-1}$ for momentum, $\sim\ 10^{-7}\ m^2.s^{-1}$ for temperature and $\sim 10^{-9}\ m^2.s^{-1}$ for salinity.

9.1.2 Richardson Number Dependent (key_zdfric)

```
!-----------------------------------------------------------------------
&namric        !   richardson number dependent vertical diffusion       ("key_zdfric" )
!-----------------------------------------------------------------------
   avmri       = 100.e-4   !  maximum value of the vertical viscosity
   alp         =   5.      !  coefficient of the parameterization
   nric        =   2       !  coefficient of the parameterization
/
```

When **key_zdfric** is defined, a local Richardson number dependent formulation for the vertical momentum and tracer eddy coefficients is set. The vertical mixing coefficients are diagnosed from the large scale variables computed by the model. *In situ* measurements have been used to link vertical turbulent activity to large scale ocean structures. The hypothesis of a mixing mainly maintained by the growth of Kelvin-Helmholtz like instabilities leads to a dependency between the vertical eddy coefficients and the local Richardson number ($i.e.$ the ratio of stratification to vertical shear). Following **?**, the following formulation has been implemented :

$$\left\{ \begin{aligned} A^{vT} &= \frac{A^{vT}_{ric}}{\left(1 + a \; Ri\right)^n} + A^{vT}_b \\ \\ A^{vm} &= \frac{A^{vT}}{\left(1 + a \; Ri\right)} + A^{vm}_b \end{aligned} \right. \tag{9.1}$$

where $Ri = N^2 / \left(\partial_z \mathbf{U}_h\right)^2$ is the local Richardson number, N is the local Brunt-Vaisälä frequency (see §4.8.2), A^{vT}_b and A^{vm}_b are the constant background values set as in the constant case (see §9.1.1), and $A^{vT}_{ric} = 10^{-4} \; m^2.s^{-1}$ is the maximum value that can be reached by the coefficient when $Ri \leq 0$, $a = 5$ and $n = 2$. The last three values can be modified by setting the *avmri*, *alp* and *nric* namelist parameters, respectively.

9.1.3 TKE Turbulent Closure Scheme (key_zdftke)

```
!-----------------------------------------------------------------------
&namtke        !   turbulent eddy kinetic dependent vertical diffusion  ("key_zdftke")
!-----------------------------------------------------------------------
   ln_rstke    = .false.   !  restart with tke from a run without tke (T) or not (F)
   nn_itke     =  50       !  number of iterative loops if ln_rstke=T
   rn_ediff    =   0.1     !  coef. for vertical eddy coef. (avt=rn_ediff*mxl*sqrt(e) )
   rn_ediss    =   0.7     !  coef. of the Kolmogoroff dissipation
   rn_ebb      =   3.75    !  coef. of the surface input of tke
   rn_efave    =   1.      !  boost of the tke diffusion ( avtke=rn_efave*avm )
   rn_emin     =   1.e-6   !  minimum value of tke [m2/s2]
   rn_emin0    =   1.e-4   !  surface minimum value of tke [m2/s2]
   nn_mxl      =   2       !  mixing length: = 0 bounded by the distance to surface and bottom
                           !                 = 1 bounded by the local vertical scale factor
                           !                 = 2 first vertical derivative of mixing length bounded by 1
                           !                 = 3 same criteria as case 2 but applied in a different way
   nn_pdl      =   1       !  Prandtl number function of richarson number (=1, avt=pdl(Ri)*avm) or not (=0, avt=avm)
   nn_avb      =   0       !  profile for constant background used on avt & avm (=1) or not (=0)
   nn_ave      =   1       !  horizontal averaged on avt (=1) or not (=0)
   ln_mxl0     = .false.   !  mixing length scale surface value as function of wind stress (T) or not (F)
   rn_lmin     =   0.4     !  interior buoyancy lenght scale minimum value
   rn_lmin0    =   0.4     !  surface  buoyancy lenght scale minimum value
   nn_etau     =   0       !  exponentially deceasing penetration of tke due to internal & intertial waves
                           !        = 0 no penetration ( O(2 km) resolution)
                           !        = 1 additional tke source
                           !        = 2 additional tke source applied only at the base of the mixed layer
   nn_htau     =   2       !  type of exponential decrease of tke penetration
                           !        = 0  constant 10 m length scale
                           !        = 1  ???
                           !        = 2  ???
   rn_efr      =   0.05    !  fraction of surface tke value which penetrates inside the ocean
   ln_lc       = .false.   !  Langmuir cell effect
   rn_lc       =   0.15    !  coef. associated to Langmuir cells
   nn_havtb    =   0       !  horizontal shape for avtb (=1) or not (=0)
/
```

The vertical eddy viscosity and diffusivity coefficients are computed from a TKE turbulent closure model based on a prognostic equation for $\bar{e}$, the turbulent kinetic energy, and a closure assumption for the turbulent length scales. This turbulent closure model has been developed by **?** in the atmospheric case, adapted by **?** for the oceanic case, and embedded in OPA by **?** for equatorial Atlantic simulations. Since then, significant modifications have been introduced by **?** in both the implementation and the formulation of the mixing length scale. The time evolution of $\bar{e}$ is the result of the production of $\bar{e}$ through vertical shear, its destruction through stratification, its vertical diffusion, and its dissipation of **?** type :

$$\frac{\partial \bar{e}}{\partial t} = \frac{A^{vm}}{e_3} \left[\left(\frac{\partial u}{\partial k} \right)^2 + \left(\frac{\partial v}{\partial k} \right)^2 \right] - A^{vT} N^2 + \frac{1}{e_3} \frac{\partial}{\partial k} \left[\frac{A^{vm}}{e_3} \frac{\partial \bar{e}}{\partial k} \right] - c_\epsilon \frac{\bar{e}^{3/2}}{l_\epsilon} \tag{9.2}$$

$$\begin{aligned} A^{vm} &= C_k\, l_k\, \sqrt{\bar{e}} \\ A^{vT} &= A^{vm} / P_{rt} \end{aligned} \tag{9.3}$$

where N is the local Brunt-Vaisälä frequency (see §4.8.2), l_ϵ and l_κ are the dissipation and mixing length scales, P_{rt} is the Prandtl number. The constants $C_k = \sqrt{2}/2$ and $C_\epsilon = 0.1$ are designed to deal with vertical mixing at any depth [**?**]. They are set through namelist parameters *ediff* and *ediss*. P_{rt} can be set to unity or, following **?**, be a function of the local Richardson number, R_i :

$$P_{rt} = \begin{cases} 1 & \text{if } R_i \leq 0.2 \\ 5\, R_i & \text{if } 0.2 \leq R_i \leq 2 \\ 10 & \text{if } 2 \leq R_i \end{cases}$$

Note that a horizontal Shapiro filter can optionally be applied to R_i. However it is an obsolescent option that is not recommended. The choice of P_{rt} is controlled by the *npdl* namelist parameter.

For computational efficiency, the original formulation of the turbulent length scales proposed by **?** has been simplified. Four formulations are proposed, the choice of which is controlled by the *nmxl* namelist parameter. The first two are based on the following first order approximation [**?**] :

$$l_k = l_\epsilon = \sqrt{2\bar{e}} / N \tag{9.4}$$

which is valid in a stable stratified region with constant values of the brunt- Vaisälä frequency. The resulting length scale is bounded by the distance to the surface or to the bottom (*nmxl*=0) or by the local vertical scale factor (*nmxl*=1). **?** notice that this simplification has two major drawbacks : it makes no sense for locally unstable stratification and the computation no longer uses all the information contained in the vertical density profile. To overcome these drawbacks, **?** introduces the *nmxl*=2 or 3 cases, which add an extra assumption concerning the vertical gradient of the computed length scale. So, the length scales are first evaluated as in (9.4) and then bounded such that :

$$\frac{1}{e_3} \left| \frac{\partial l}{\partial k} \right| \leq 1 \qquad \text{with } \; l = l_k = l_\epsilon \tag{9.5}$$

(9.5) means that the vertical variations of the length scale cannot be larger than the variations of depth. It provides a better approximation of the **?** formulation while being much less time consuming. In particular, it allows the length scale to be limited not only by

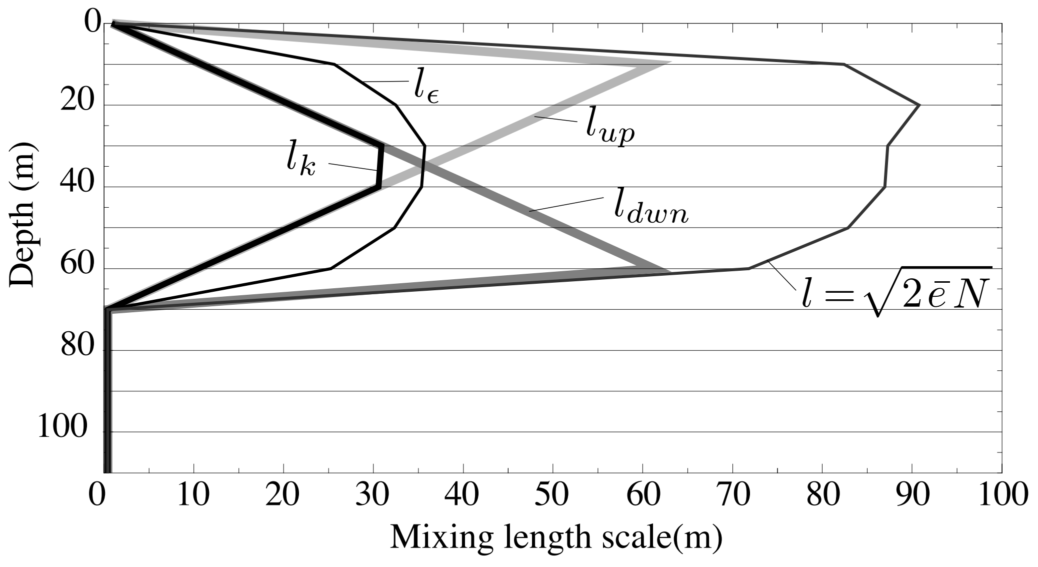


FIG. 9.1 – Illustration of the mixing length computation.

the distance to the surface or to the ocean bottom but also by the distance to a strongly stratified portion of the water column such as the thermocline (Fig. 9.1.3). In order to impose the (9.5) constraint, we introduce two additional length scales : l_{up} and l_{dwn}, the upward and downward length scales, and evaluate the dissipation and mixing turbulent length scales as (and note that here we use numerical indexing) :

$$\begin{aligned} l_{up}^{(k)} &= \min\left(l^{(k)} \ , \ l_{up}^{(k+1)} + e_{3T}^{(k)} \quad \right) && \text{from } k=1 \text{ to } jpk \\ l_{dwn}^{(k)} &= \min\left(l^{(k)} \ , \ l_{dwn}^{(k-1)} + e_{3T}^{(k-1)} \right) && \text{from } k=jpk \text{ to } 1 \end{aligned} \tag{9.6}$$

where $l^{(k)}$ is computed using (9.4), $i.e.$ $l^{(k)} = \sqrt{2\bar{e}^{(k)}/N^{(k)}}$.

In the *nmxl*=2 case, the dissipation and mixing length scales take the same value : $l_k = l_\epsilon = \min\left(\ l_{up} \ , \ l_{dwn} \ \right)$, while in the *nmxl*=2 case, the dissipation and mixing length scales are give as in **?** :

$$\begin{aligned} l_k &= \sqrt{\ l_{up} \ \ l_{dwn}\ } \\ l_\epsilon &= \min\left(\ l_{up} \ , \ l_{dwn} \ \right) \end{aligned} \tag{9.7}$$

At the sea surface the value of $\bar{e}$ is prescribed from the wind stress field : $\bar{e} = ebb \ |\tau|$ ($ebb = 60$ by default) with a minimal threshold of $emin0 = 10^{-4} \ m^2.s^{-2}$ (namelist parameters). Its value at the bottom of the ocean is assumed to be equal to the value of the level just above. The time integration of the $\bar{e}$ equation may formally lead to negative values because the numerical scheme does not ensure its positivity. To overcome this problem, a cut-off in the minimum value of $\bar{e}$ is used. Following **?**, the cut-off value is set to $\sqrt{2}/2\ 10^{-6}\ m^2.s^{-2}$. This allows the subsequent formulations to match that of**?** for the diffusion in the thermocline and deep ocean : $(A^{vT} = 10^{-3}/N)$. In addition, a cut-off is applied on A^{vm} and A^{vT} to avoid numerical instabilities associated with too weak vertical diffusion. They must be specified at least larger than the molecular values, and are set through *avm0* and *avt0* (**namelist** parameters).

9.1.4 K Profile Parametrisation (KPP) (key_zdfkpp)

```
!-----------------------------------------------------------------------
&namkpp        !   K-Profile Parameterization dependent vertical mixing  ("key_zdfkpp", and optionnaly:
!                                                                         "key_kppcustom" or "key_kpplktb")
!-----------------------------------------------------------------------
   ln_kpprimix = .true.    !  shear instability mixing
   difmiw      =  1.0e-04  !  constant internal wave viscosity [m2/s]
   difsiw      =  0.1e-04  !  constant internal wave diffusivity [m2/s]
   Riinfty     =  0.8      !  local Richardson Number limit for shear instability
   difri       =  0.0050   !  maximum shear mixing at Rig = 0    [m2/s]
   bvsqcon     = -0.01e-07 !  Brunt-Vaisala squared for maximum convection [1/s2]
   difcon      =  1.       !  maximum mixing in interior convection [m2/s]
   navb        =  0        !  horizontal averaged (=1) or not (=0) on avt and amv
   nave        =  1        !  constant (=0) or profile (=1) background on avt
/
```

The K-Profile Parametrization (KKP) developed by **?** has been implemented in *NEMO* by J. Chanut (PhD reference to be added here !).

Add a description of KPP here.

9.2 Convection

```
!-----------------------------------------------------------------------
&namzdf        !   vertical physics
!-----------------------------------------------------------------------
   avm0        =   1.2e-4  !  vertical eddy viscosity   [m2/s]          (background Kz if not "key_zdfcst")
   avt0        =   1.2e-5  !  vertical eddy diffusivity [m2/s]          (background Kz if not "key_zdfcst")
   ln_zdfnpc   = .false.   !  convection: Non-Penetrative algorithm (T) or not (F)
   ln_zdfevd   = .true.    !  convection: enhanced vertical diffusion (T) or not (F)
   avevd       = 100.      !  vertical coefficient for enhanced diffusion scheme [m2/s]
   n_evdm      =   1       !  enhanced mixing apply on tracer (=0) or on tracer and momentum (=1)
   ln_zdfexp   =  .false.  !  split explicit (T) or implicit (F) time stepping
   n_zdfexp    =   3       !  number of sub-timestep for ln_zdfexp=T
/
```

Static instabilities (i.e. light potential densities under heavy ones) may occur at particular ocean grid points. In nature, convective processes quickly re-establish the static stability of the water column. These processes have been removed from the model via the hydrostatic assumption so they must be parameterized. Three parameterisations are available to deal with convective processes : a non-penetrative convective adjustment or an enhanced vertical diffusion, or/and the use of a turbulent closure scheme.

9.2.1 Non-Penetrative Convective Adjustment (*ln_tranpc*=.true.)

```
!-----------------------------------------------------------------------
&namnpc        !   non penetrative convection
!-----------------------------------------------------------------------
   nnpc1       =    1      !  non penetrative convective scheme computation frequency
   nnpc2       =  365      !  non penetrative convective scheme print frequency
/
```

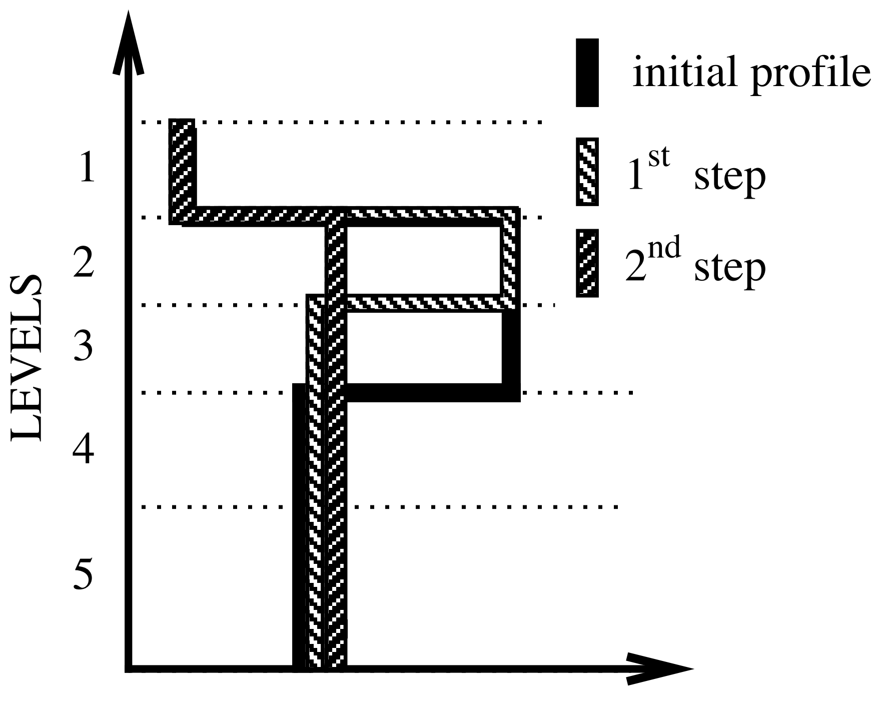


FIG. 9.2 – Example of an unstable density profile treated by the non penetrative convective adjustment algorithm. 1^{st} step : the initial profile is checked from the surface to the bottom. It is found to be unstable between levels 3 and 4. They are mixed. The resulting ρ is still larger than $\rho(5)$: levels 3 to 5 are mixed. The resulting ρ is still larger than $\rho(6)$: levels 3 to 6 are mixed. The 1^{st} step ends since the density profile is then stable below the level 3. 2^{nd} step : the new ρ profile is checked following the same procedure as in 1^{st} step : levels 2 to 5 are mixed. The new density profile is checked. It is found stable : end of algorithm.

The non-penetrative convective adjustment is used when *ln_zdfnpc*=true. It is applied at each *nnpc1* time step and mixes downwards instantaneously the statically unstable portion of the water column, but only until the density structure becomes neutrally stable ($i.e.$ until the mixed portion of the water column has *exactly* the density of the water just below) [**?**]. The associated algorithm is an iterative process used in the following way (Fig. 9.2.1) : starting from the top of the ocean, the first instability is found. Assume in the following that the instability is located between levels k and $k+1$. The potential temperature and salinity in the two levels are vertically mixed, conserving the heat and salt contents of the water column. The new density is then computed by a linear approximation. If the new density profile is still unstable between levels $k+1$ and $k+2$, levels k, $k+1$ and $k+2$ are then mixed. This process is repeated until stability is established below the level k (the mixing process can go down to the ocean bottom). The algorithm is repeated to check if the density profile between level $k-1$ and k is unstable and/or if there is no deeper instability.

This algorithm is significantly different from mixing statically unstable levels two by two. The latter procedure cannot converge with a finite number of iterations for some vertical profiles while the algorithm used in *NEMO* converges for any profile in a number of iterations which is less than the number of vertical levels. This property is of paramount importance as pointed out by **?** : it avoids the existence of permanent and unrealistic static instabilities at the sea surface. This non-penetrative convective algorithm has been proved successful in studies of the deep water formation in the north-western Mediterranean Sea [**???**].

Note that in the current implementation of this algorithm presents several limitations. First, potential density referenced to the sea surface is used to check whether the density profile is stable or not. This is a strong simplification which leads to large errors for realistic ocean simulations. Indeed, many water masses of the world ocean, especially Antarctic Bottom Water, are unstable when represented in surface-referenced potential density. The scheme will erroneously mix them up. Second, the mixing of potential density is assumed to be linear. This assures the convergence of the algorithm even when the equation of state is non-linear. Small static instabilities can thus persist due to cabbeling : they will be treated at the next time step. Third, temperature and salinity, and thus density, are mixed, but the corresponding velocity fields remain unchanged. When using a Richardson Number dependent eddy viscosity, the mixing of momentum is done through the vertical diffusion : after a static adjustment, the Richardson Number is zero and thus the eddy viscosity coefficient is at a maximum. When this convective adjustment algorithm is used with constant vertical eddy viscosity, spurious solutions can occur since the vertical momentum diffusion remains small even after a static adjustment. In that case, we recommend the addition of momentum mixing in a manner that mimics the mixing in temperature and salinity [**??**].

9.2.2 Enhanced Vertical Diffusion (*ln_zdfevd*=.true.)

```
!-----------------------------------------------------------------------
&namzdf        !   vertical physics
!-----------------------------------------------------------------------
   avm0        =   1.2e-4  !  vertical eddy viscosity   [m2/s]          (background Kz if not "key_zdfcst")
   avt0        =   1.2e-5  !  vertical eddy diffusivity [m2/s]          (background Kz if not "key_zdfcst")
   ln_zdfnpc   = .false.   !  convection: Non-Penetrative algorithm (T) or not (F)
   ln_zdfevd   = .true.    !  convection: enhanced vertical diffusion (T) or not (F)
   avevd       = 100.      !  vertical coefficient for enhanced diffusion scheme [m2/s]
   n_evdm      =   1       !  enhanced mixing apply on tracer (=0) or on tracer and momentum (=1)
   ln_zdfexp   =  .false.  !  split explicit (T) or implicit (F) time stepping
   n_zdfexp    =   3       !  number of sub-timestep for ln_zdfexp=T
/
```

The enhanced vertical diffusion parameterisation is used when *ln_zdfevd*=true. In this case, the vertical eddy mixing coefficients are assigned very large values (a typical value is $10\ m^2 s^{-1})$ in regions where the stratification is unstable ($i.e.$ when the Brunt-Vaisälä frequency is negative) [**??**]. This is done either on tracers only (*n_evdm*=0) or on both momentum and tracers (*n_evdm*=1).

In practice, where $N^2 \leq 10^{-12}$, A_T^{vT} and A_T^{vS}, and if *n_evdm*=1, the four neighbouring A_u^{vm} and A_v^{vm} values also, are set equal to the namelist parameter *avevd*. A typical value for $avevd$ is between 1 and 100 $m^2.s^{-1}$. This parameterisation of convective processes is less time consuming than the convective adjustment algorithm presented above when mixing both tracers and momentum in the case of static instabilities. It requires the use of an implicit time stepping on vertical diffusion terms (i.e. *ln_zdfexp*=false).

9.2.3 Turbulent Closure Scheme (key_zdftke)

The TKE turbulent closure scheme presented in §9.1.3 and used when the **key_zdftke** is defined, in theory solves the problem of statically unstable density profiles. In such a case, the term corresponding to the destruction of turbulent kinetic energy through stratification in (9.2) becomes a source term, since N^2 is negative. It results in large values of A_T^{vT} and A_T^{vT}, and also the four neighbouring $A_u^{vm} and\ A_v^{vm}$ (up to 1 $m^2 s^{-1}$). These large values restore the static stability of the water column in a way similar to that of the enhanced vertical diffusion parameterisation (§9.2.2). However, in the vicinity of the sea surface (first ocean layer), the eddy coefficients computed by the turbulent closure scheme do not usually exceed $10^{-2} m.s^{-1}$, because the mixing length scale is bounded by the distance to the sea surface (see §9.1.3). It can thus be useful to combine the enhanced vertical diffusion with the turbulent closure scheme, $i.e.$ setting the *ln_zdfnpc* namelist parameter to true and defining the **key_zdftke** CPP key all together.

The KPP turbulent closure scheme already includes enhanced vertical diffusion in the case of convection, as governed by the variables $bvsqcon$ and $difcon$ found in *zdfkpp.F90*, therefore *np_zdfevd* should not be used with the KPP scheme.

9.3 Double Diffusion Mixing (*zdfddm.F90* module - key_zdfddm)

```
!-----------------------------------------------------------------------
&namddm        !   double diffusive mixing parameterization              ("key_zdfddm")
!-----------------------------------------------------------------------
     avts     = 1.e-4     !  maximum avs (vertical mixing on salinity)
     hsbfr    = 1.6       !  heat/salt buoyancy flux ratio
/
```

Double diffusion occurs when relatively warm, salty water overlies cooler, fresher water, or vice versa. The former condition leads to salt fingering and the latter to diffusive convection. Double-diffusive phenomena contribute to diapycnal mixing in extensive regions of the ocean. **?** include a parameterisation of such phenomena in a global ocean model and show that it leads to relatively minor changes in circulation but exerts significant regional influences on temperature and salinity.

Diapycnal mixing of S and T are described by diapycnal diffusion coefficients

$$A^{vT} = A_o^{vT} + A_f^{vT} + A_d^{vT}$$
$$A^{vS} = A_o^{vS} + A_f^{vS} + A_d^{vS}$$

where subscript f represents mixing by salt fingering, d by diffusive convection, and o by processes other than double diffusion. The rates of double-diffusive mixing depend on the buoyancy ratio $R_\rho = \alpha \partial_z T / \beta \partial_z S$, where α and β are coefficients of thermal expansion

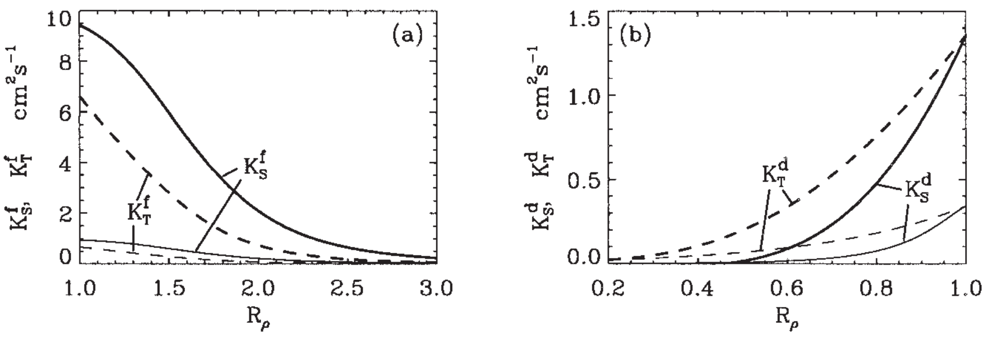


FIG. 9.3 – From **?** : (a) Diapycnal diffusivities A_f^{vT} and A_f^{vS} for temperature and salt in regions of salt fingering. Heavy curves denote $A^{*v} = 10^{-3}\ m^2.s^{-1}$ and thin curves $A^{*v} = 10^{-4}\ m^2.s^{-1}$; (b) diapycnal diffusivities A_d^{vT} and A_d^{vS} for temperature and salt in regions of diffusive convection. Heavy curves denote the Federov parameterisation and thin curves the Kelley parameterisation. The latter is not implemented in *NEMO* .

and saline contraction (see §4.8.1). To represent mixing of S and T by salt fingering, we adopt the diapycnal diffusivities suggested by Schmitt (1981) :

$$A_f^{vS} = \begin{cases} \frac{A^{*v}}{1+(R_\rho/R_c)^n} & \text{if } R_\rho > 1 \text{ and } N^2 > 0 \\ 0 & \text{otherwise} \end{cases} \tag{9.8}$$

$$A_f^{vT} = 0.7\ A_f^{vS}/R_\rho \tag{9.9}$$

The factor 0.7 in (9.9) reflects the measured ratio $\alpha F_T/\beta F_S \approx 0.7$ of buoyancy flux of heat to buoyancy flux of salt ($e.g.$, **?**). Following **?**, we adopt $R_c = 1.6$, $n = 6$, and $A^{*v} = 10^{-4}\ m^2.s^{-1}$.

To represent mixing of S and T by diffusive layering, the diapycnal diffusivities suggested by Federov (1988) is used :

$$A_d^{vT} = \begin{cases} 1.3635\ \exp\left(4.6\ \exp\left[-0.54\left(R_\rho^{-1} - 1\right)\right]\right) & \text{if } 0 < R_\rho < 1 \text{ and } N^2 > 0 \\ 0 & \text{otherwise} \end{cases} \tag{9.10}$$

$$A_d^{vS} = \begin{cases} A_d^{vT}\ (1.85\,R_\rho - 0.85) & \text{if } 0.5 \leq R_\rho < 1 \text{ and } N^2 > 0 \\ A_d^{vT}\ 0.15\ R_\rho & \text{if } \ 0 < R_\rho < 0.5 \text{ and } N^2 > 0 \\ 0 & \text{otherwise} \end{cases} \tag{9.11}$$

The dependencies of (9.8) to (9.11) on R_ρ are illustrated in Fig. 9.3. Implementing this requires computing R_ρ at each grid point on every time step. This is done in *eosbn2.F90* at the same time as N^2 is computed. This avoids duplication in the computation of α and β (which is usually quite expensive).

9.4 Bottom Friction (*zdfbfr.F90* module)

```
!-----------------------------------------------------------------------
&nambfr        !   bottom friction
!-----------------------------------------------------------------------
   nbotfr      =    1      !  type of bottom friction :   = 0 : no   slip,  = 2 : nonlinear friction
                           !                              = 3 : free slip,  = 1 :    linear friction
   bfri1       =    4.e-4  !  bottom drag coefficient (linear case)
   bfri2       =    1.e-3  !  bottom drag coefficient (non linear case)
   bfeb2       =    2.5e-3 !  bottom turbulent kinetic energy background  (m^2/s^2)
/
```

Both the surface momentum flux (wind stress) and the bottom momentum flux (bottom friction) enter the equations as a condition on the vertical diffusive flux. For the bottom boundary layer, one has :

$$A^{vm}\left(\partial \mathbf{U}_h / \partial z\right) = \mathbf{F}_h \tag{9.12}$$

where $\mathbf{F}_h$ is supposed to represent the horizontal momentum flux outside the logarithmic turbulent boundary layer (thickness of the order of 1 m in the ocean). How $\mathbf{F}_h$ influences the interior depends on the vertical resolution of the model near the bottom relative to the Ekman layer depth. For example, in order to obtain an Ekman layer depth $d = \sqrt{2\ A^{vm}}/f = 50$ m, one needs a vertical diffusion coefficient $A^{vm} = 0.125$ m^{2}s^{-1} (for a Coriolis frequency $f = 10^{-4}$ m^{2}s^{-1}). With a background diffusion coefficient $A^{vm} = 10^{-4}$ m^{2}s^{-1}, the Ekman layer depth is only 1.4 m. When the vertical mixing coefficient is this small, using a flux condition is equivalent to entering the viscous forces (either wind stress or bottom friction) as a body force over the depth of the top or bottom model layer. To illustrate this, consider the equation for u at k, the last ocean level :

$$\frac{\partial u\ (k)}{\partial t} = \frac{1}{e_{3u}}\left[A^{vm}\ (k)\frac{U\ (k-1) - U\ (k)}{e_{3uw}\ (k-1)} - F_u\right] \approx -\frac{F_u}{e_{3u}} \tag{9.13}$$

For example, if the bottom layer thickness is 200 m, the Ekman transport will be distributed over that depth. On the other hand, if the vertical resolution is high (1 m or less) and a turbulent closure model is used, the turbulent Ekman layer will be represented explicitly by the model. However, the logarithmic layer is never represented in current primitive equation model applications : it is *necessary* to parameterize the flux $\mathbf{F}_h$. Two choices are available in *NEMO* : a linear and a quadratic bottom friction. Note that in both cases, the rotation between the interior velocity and the bottom friction is neglected in the present release of *NEMO* .

9.4.1 Linear Bottom Friction (*nbotfr* = 1)

The linear bottom friction parameterisation assumes that the bottom friction is proportional to the interior velocity (i.e. the velocity of the last model level) :

$$\mathbf{F}_h = \frac{A^{vm}}{e_3}\ \frac{\partial \mathbf{U}_h}{\partial k} = r\ \mathbf{U}_h^b \tag{9.14}$$

where $\mathbf{U}_h^b$ is the horizontal velocity vector of the bottom ocean layer and r is a friction coefficient expressed in m.s^{-1}. This coefficient is generally estimated by setting a typical decay time τ in the deep ocean, and setting $r = H/\tau$, where H is the ocean depth. Commonly accepted values of τ are of the order of 100 to 200 days [**?**]. A value $\tau^{-1} = 10^{-7}$ s^{-1} equivalent to 115 days, is usually used in quasi-geostrophic models. One may consider the linear friction as an approximation of quadratic friction, $r \approx 2\ C_D\ U_{av}$ (**?**, Eq. 9.6.6). For example, with a drag coefficient $C_D = 0.002$, a typical speed of tidal currents of $U_{av} = 0.1$ m.s^{-1}, and assuming an ocean depth $H = 4000$ m, the resulting friction coefficient is $r = 4\ 10^{-4}$ m.s^{-1}. This is the default value used in *NEMO* . It corresponds to a decay time scale of 115 days. It can be changed by specifying *bfric1* (namelist parameter).

In the code, the bottom friction is imposed by updating the value of the vertical eddy coefficient at the bottom level. Indeed, the discrete formulation of (9.14) at the last ocean $T-$level, using the fact that $\mathbf{U}_h = 0$ below the ocean floor, leads to

$$\begin{aligned} A_u^{vm} &= r\ e_{3uw} \\ A_v^{vm} &= r\ e_{3vw} \end{aligned} \tag{9.15}$$

This update is done in *zdfbfr.F90* when *nbotfr*=1. The value of r used is *bfric1*. Setting *nbotfr*=3 is equivalent to setting $r = 0$ and leads to a free-slip bottom boundary condition. Setting *nbotfr*=0 sets $r = 2\ A_{vb}^{\mathbf{U}}$, where $A_{vb}^{\mathbf{U}}$ is the background vertical eddy coefficient, and a no-slip boundary condition is imposed. Note that this latter choice generally leads to an underestimation of the bottom friction : for example with a deepest level thickness of 200 m and $A_{vb}^{\mathbf{U}} = 10^{-4}$m^2.s^{-1}, the friction coefficient is only $r = 10^{-6}$m.s^{-1}.

9.4.2 Non-Linear Bottom Friction (*nbotfr* = 2)

The non-linear bottom friction parameterisation assumes that the bottom friction is quadratic :

$$\mathbf{F}_h = \frac{A^{vm}}{e_3}\frac{\partial \mathbf{U}_h}{\partial k} = C_D\ \sqrt{u_b^2 + v_b^2 + e_b}\ \ \mathbf{U}_h^b \tag{9.16}$$

with $\mathbf{U}_h^b = (u_b\ ,\ v_b)$ the horizontal interior velocity ($i.e.$ the horizontal velocity of the bottom ocean layer), C_D a drag coefficient, and e_b a bottom turbulent kinetic energy due to tides, internal waves breaking and other short time scale currents. A typical value of the drag coefficient is $C_D = 10^{-3}$. As an example, the CME experiment [**?**] uses $C_D = 10^{-3}$ and $e_b = 2.5\ 10^{-3}$m^2.s^{-2}, while the FRAM experiment [**?**] uses $e_b = 0$ and $e_b = 2.5\ \ 10^{-3}$m^2.s^{-2}. The FRAM choices have been set as default values (*bfric2* and *bfeb2* namelist parameters).

As for the linear case, the bottom friction is imposed in the code by updating the value of the vertical eddy coefficient at the bottom level :

$$\begin{aligned} A_u^{vm} &= C_D\ e_{3uw}\left[u^2 + \left(\overline{\overline{v}}^{i+1,j}\right)^2 + e_b\right]^{1/2} \\ A_v^{vm} &= C_D\ e_{3uw}\left[\left(\overline{\overline{u}}^{i,j+1}\right)^2 + v^2 + e_b\right]^{1/2} \end{aligned} \tag{9.17}$$

This update is done in *zdfbfr.F90*. The coefficients that control the strength of the non-linear bottom friction are initialized as namelist parameters : C_D= *bfri2*, and e_b =*bfeb2*.

Miscellaneous Topics (xxx)

Contents

10.1 Representation of Unresolved Straits

10.1.1 Hand made geometry changes

• reduced scale factor, also called partially open face (Hallberg, personnal communication 2006)

• increase of the viscous boundary layer thickness by local increase of the fmask value at the coast

Add a short description of scale factor changes staff and fmask increase

10.1.2 Cross Land Advection (*tracla* module)

```
!-----------------------------------------------------------------------
&namcla        !   cross land advection
!-----------------------------------------------------------------------
   n_cla       =    1      !  advection between 2 ocean pts separates by land
/
```

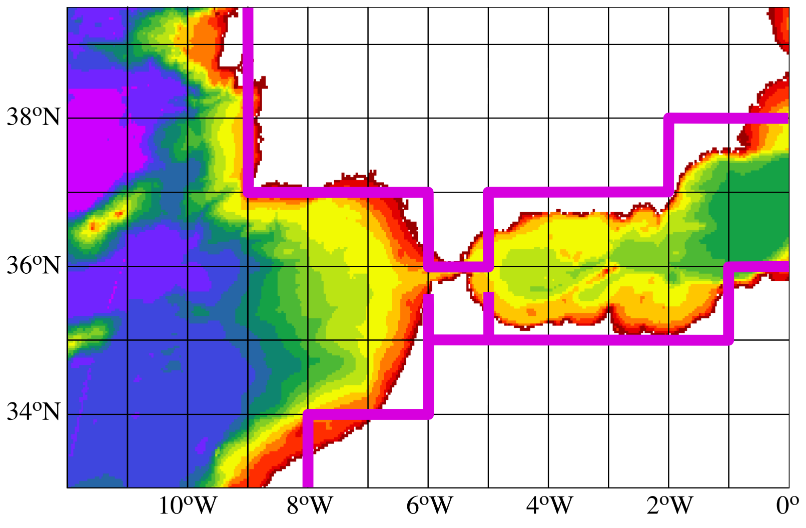
Add a short description of CLA staff here or in lateral boundary condition chapter ?

10.2 Closed seas

10.3 Sub-Domain Functionality (*jpizoom, jpjzoom*)

The sub-domain functionality, also improperly called the zoom option (improperly because it is not associated with a change in model resolution) is a quite simple function that allows a simulation over a sub-domain of an already defined configuration ($i.e.$ without defining a new mesh, initial state and forcings). This option can be useful for testing the user settings of surface boundary conditions, or the initial ocean state of a huge ocean model configuration while having a small computer memory requirement. It can also be used to easily test specific physics in a sub-domain (for example, see [**?**] for a test of the coupling used in the global ocean version of OPA between sea-ice and ocean model over the Arctic or Antarctic ocean, using a sub-domain). In the standard model, this option does not include any specific treatment for the ocean boundaries of the sub-domain : they are considered as artificial vertical walls. Nevertheless, it is quite easy to add a restoring term toward a climatology in the vicinity of such boundaries (see §4.6).

In order to easily define a sub-domain over which the computation can be performed, the dimension of all input arrays (ocean mesh, bathymetry, forcing, initial state, ...) are defined as *jpidta*, *jpjdta* and *jpkdta* (*par_oce.F90* module), while the computational domain is defined through *jpiglo*, *jpjglo* and *jpk* (*par_oce.F90* module). When running the model over the whole domain, the user sets *jpiglo=jpidta jpjglo=jpjdta* and *jpk=jpkdta*. When running the model over a sub-domain, the user has to provide the size of the sub-domain, (*jpiglo, jpjglo, jpkglo*), and the indices of the south western corner as *jpizoom* and *jpjzoom* in the *par_oce.F90* module (Fig. 10.3).



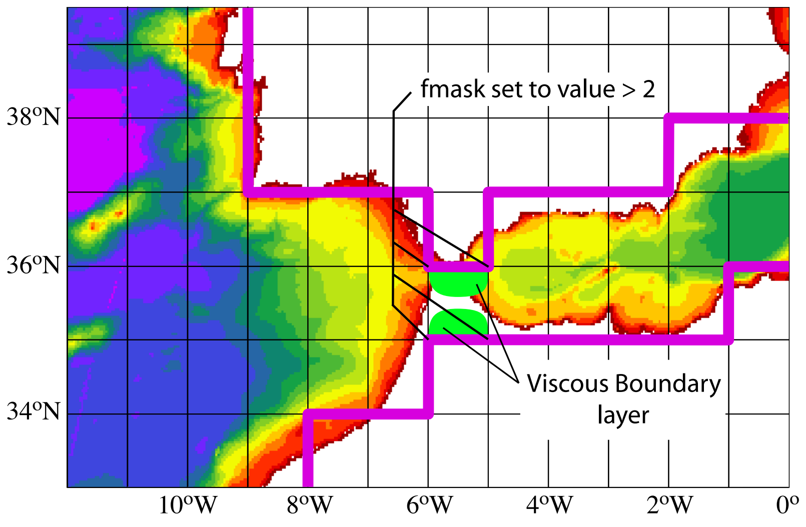


FIG. 10.1 – Example of the Gibraltar strait defined in a 1° x 1° mesh. *Top* : using partially open cells. The meridional scale factor at v-point is reduced on both sides of the strait to account for the real width of the strait (about 20 km). Note that the scale factors of the strait T-point remains unchanged. *Bottom* : using viscous boundary layers. The four fmask parameters along the strait coastlines are set to a value larger than 4, $i.e.$ "strong" no-slip case (see Fig.7.1) creating a large viscous boundary layer that allows a reduced transport through the strait.

Note that a third set of dimensions exist, *jpi*, *jpj* and *jpk* which is actually used to perform the computation. It is set by default to *jpi=jpjglo* and *jpj=jpjglo*, except for massively parallel computing where the computational domain is laid out on local processor memories following a 2D horizontal splitting.

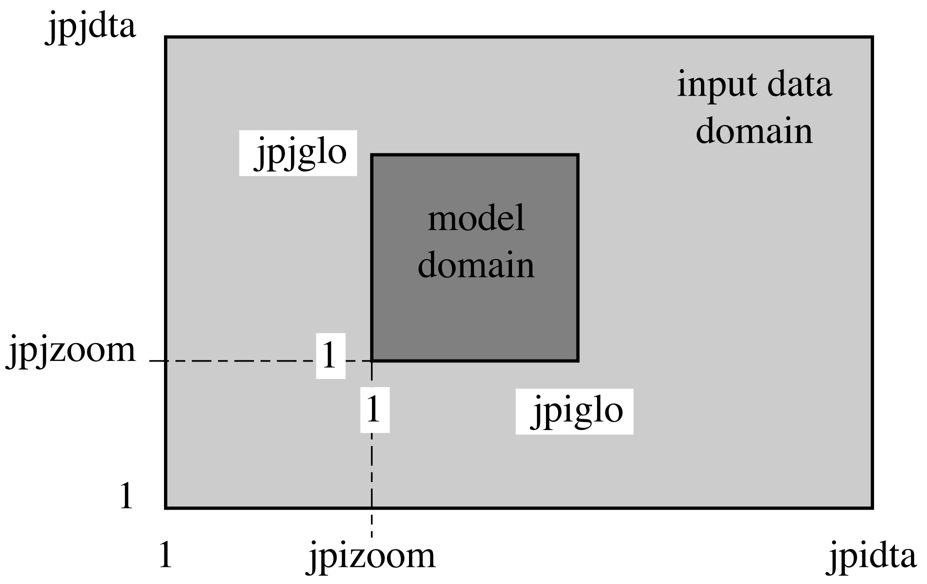


FIG. 10.2 – Position of a model domain compared to the data input domain when the zoom functionality is used.

10.4 Water column model : 1D model (key_cfg_1d)

The 1D model option simulates a stand alone water column within the 3D *NEMO* system. It can be applied to the ocean alone or to the ocean-ice system and can include passive tracers or a biogeochemical model. It is set up by defining the **key_cfg_1d** CPP key. This 1D model is a very useful tool *(a)* to learn about the physics and numerical treatment of vertical mixing processes ; *(b)* to investigate suitable parameterisations of unresolved turbulence (wind steering, langmuir circulation, skin layers) ; *(c)* to compare the behaviour of different vertical mixing schemes ; *(d)* to perform sensitivity studies on the vertical diffusion at a particular point of the ocean global domain ; *(d)* to produce extra diagnostics, without the large memory requirement of the full 3D model.

The methodology is based on the use of the zoom functionality (see §10.3), with some extra routines. There is no need to define a new mesh, bathymetry, initial state or forcing, since the 1D model will use those of the configuration it is a zoom of.

10.5 Accelerating the Convergence (*nn_acc* = 1)

```
!-----------------------------------------------------------------------
&namdom        !   space and time domain (bathymetry, mesh, timestep)
!-----------------------------------------------------------------------
   ntopo       =    1      !  compute (=0) or read(=1) the bathymetry file
   e3zps_min   =    5.     !  the thickness of the partial step is set larger than the minimum
   e3zps_rat   =    0.1    !  of e3zps_min and e3zps_rat * e3t   (N.B. 0<e3zps_rat<1)
   nmsh        =    0      !  create (=1) a mesh file (coordinates, scale factors, masks) or not (=0)
   nacc        =    0      !  =1 acceleration of convergence method used, rdt < rdttra(k)
                           !  =0, no acceleration, rdt = rdttra
   atfp        =    0.1    !  asselin time filter parameter
   rdt         = 5760.     !  time step for the dynamics (and tracer if nacc=0)
   rdtmin      = 5760.     !  minimum time step on tracers (used if nacc=1)
   rdtmax      = 5760.     !  maximum time step on tracers (used if nacc=1)
   rdth        =  800.     !  depth variation of tracer time step  (used if nacc=1)
   rdtbt       =   90.     !  barotropic time step (for the split explicit algorithm) ("key_dynspg_ts")
   nclosea     =    0      !  = 0 no closed sea in the model domain
                           !  = 1 closed sea (Black Sea, Caspian Sea, Great US Lakes...)
/
```

Searching an equilibrium state with an ocean model requires very long time integration (a few thousand years for a global model). Due to the size of the time step required for numerical stability (less than a few hours), this usually requires a large elapsed time. In order to overcome this problem, **?** introduces a technique that is intended to accelerate the spin up to equilibrium. It uses a larger time step in the thermodynamic evolution equations than in the dynamic evolution equations. It does not affect the equilibrium solution but modifies the trajectory to reach it.

The acceleration of convergence option is used when *nn_acc*=1. In that case, $\Delta t = rdt$ is the time step of dynamics while $\widetilde{\Delta t} = rdttra$ is the tracer time-step. Their values are set from the *rdt* and *rdttra* namelist parameters. The set of prognostic equations to solve becomes :

$$\begin{aligned} \frac{\partial \mathbf{U}_h}{\partial t} &\equiv \frac{\mathbf{U}_h^{t+1} - \mathbf{U}_h^{t-1}}{2\Delta t} = \ldots \\ \frac{\partial T}{\partial t} &\equiv \frac{T^{t+1} - T^{t-1}}{2\widetilde{\Delta t}} = \ldots \\ \frac{\partial S}{\partial t} &\equiv \frac{S^{t+1} - S^{t-1}}{2\widetilde{\Delta t}} = \ldots \end{aligned} \tag{10.1}$$

? has examined the consequences of this distorted physics. Free waves have a slower phase speed, their meridional structure is slightly modified, and the growth rate of baroclinically unstable waves is reduced but with a wider range of instability. This technique is efficient for searching for an equilibrium state in coarse resolution models. However its application is not suitable for many oceanic problems : it cannot be used for transient or time evolving problems (in particular, it is very questionable to use this technique when there is a seasonal cycle in the forcing fields), and it cannot be used in high-resolution models where baroclinically unstable processes are important. Moreover, the vertical variation of $\widetilde{\Delta t}$ implies that the heat and salt contents are no longer conserved due to the vertical coupling of the ocean level through both advection and diffusion.

10.6 Model Optimisation, Control Print and Benchmark

```
!-----------------------------------------------------------------------
```

```
&namctl       !   Control prints & Benchmark
!-----------------------------------------------------------------------
   ln_ctl     = .false.    !  trends control print (expensive!)
   nprint     =    0       !  level of print (0 no extra print)
   nictls     =    0       !  start i indice of control sum (use to compare mono versus
   nictle     =    0       !  end   i indice of control sum        multi processor runs
   njctls     =    0       !  start j indice of control               over a subdomain)
   njctle     =    0       !  end   j indice of control
   isplt      =    1       !  number of processors in i-direction
   jsplt      =    1       !  number of processors in j-direction
   nbench     =    0       !  Bench mode (1/0): CAUTION use zero except for bench
                           !     (no physical validity of the results)
   nbit_cmp   =    0       !  bit comparison mode (1/0): CAUTION use zero except for test
                           !     of comparison between single and multiple processor runs
/
```

Three issues to be described here :

• Vector and memory optimisation :

key_vectopt_loop enables internal loop collapse, a very efficient way to increase the length of vector calculations and thus speed up the model on vector computers.

key_vectopt_memory has been introduced in order to reduce the memory requirement of the model. This is obviously done at the cost of increasing the CPU time requirement, since it suppress intermediate computations which would have been saved in in-core memory. This feature has not been intensively used. In fact, currently it is only implemented for the TKE physics, in which, when **key_vectopt_memory** is defined, the coefficients used for horizontal smoothing of A_v^T and A_v^m are no longer computed once and for all. This reduces the memory requirement by three 2D arrays.

• Control print

1- *ln_ctl* : compute and print the trends averaged over the interior domain in all TRA, DYN, LDF and ZDF modules. This option is very helpful when diagnosing the origin of an undesired change in model results.

2- also *ln_ctl* but using the nictl and njctl namelist parameters to check the source of differences between mono and multi processor runs.

3- **key_esopa** (to be rename key_nemo) : which is another option for model management. When defined, this key forces the activation of all options and CPP keys. For example, all tracer and momentum advection schemes are called ! There is therefore no physical meaning associated with the model results. However, this option forces both the compiler and the model to run through all the FORTRAN lines of the model. This allows the user to check for obvious compilation or execution errors with all CPP options, and errors in namelist options.

3- **key_esopa** (to be rename key_nemo) : which is another option for model management. When defined, this key forces the activation of all options and CPP keys. For example, all tracer and momentum advection schemes are called ! There is therefore no physical meaning associated with the model results. However, this option forces both the compiler and the model to run through all the Fortran lines of the model. This allows the user to check for obvious compilation or execution errors with all CPP options, and errors in namelist options.

4- last digit comparison (*nbit_cmp*). In an MPP simulation, the computation of a sum over the whole domain is performed as the summation over all processors of each of their sums over their interior domains. This double sum never gives exactly the same result as a single sum over the whole domain, due to truncation differences. The ”bit comparison” option has been introduced in order to be able to check that mono-processor and multi-processor runs give exactly the same results.

• Benchmark (*nbench*). This option defines a benchmark run based on a GYRE configuration in which the resolution remains the same whatever the domain size. This allows a very large model domain to be used, just by changing the domain size (*jpiglo*, *jpjglo*) and without adjusting either the time-step or the physical parameterisations.

10.7 Elliptic solvers (SOL directory)

```
!-----------------------------------------------------------------------
&namsol       !   elliptic solver / island / free surface
!-----------------------------------------------------------------------
   nsolv      =      1    !  elliptic solver: =1 preconditioned conjugate gradient (pcg)
                          !                   =2 successive-over-relaxation (sor)
                          !                   =3 FETI (fet)                           ("key_feti")
                          !                   =4 sor with extra outer halo
   nsol_arp   =      0    !  absolute/relative (0/1) precision convergence test
   nmin       =    300    !  minimum of iterations for the SOR solver
   nmax       =    800    !  maximum of iterations for the SOR solver
   nmod       =     10    !  frequency of test for the SOR solver
   eps        =  1.e-6    !  absolute precision of the solver
   resmax     =  1.e-10   !  absolute precision for the SOR solver
   sor        =  1.92     !  optimal coefficient for SOR solver (to be adjusted with the domain)
   epsisl     =  1.e-10   !  absolute precision on stream function solver
   nmisl      =   4000    !  maximum pcg iterations for island                    ("key_islands")
   rnu        =      1.   !  strength of the additional force used in filtered free surface
/
```

The computation of the surface pressure gradient with a rigid lid assumption requires the computation of $\partial_t \psi$, the time evolution of the barotropic streamfunction. $\partial_t \psi$ is the solution of an elliptic equation (2.12) for which three solvers are available : a Successive-Over-Relaxation scheme (SOR), a preconditioned conjugate gradient scheme(PCG) [**??**] and a Finite Elements Tearing and Interconnecting scheme (FETI) [**??**]. The PCG is a very efficient method for solving elliptic equations on vector computers. It is a fast and rather easy method to use ; which are attractive features for a large number of ocean situations (variable bottom topography, complex coastal geometry, variable grid spacing, islands, open or cyclic boundaries, etc ...). It does not require a search for an optimal parameter as in the SOR method. However, the SOR has been retained because it is a linear solver, which is a very useful property when using the adjoint model of *NEMO* . The FETI solver is a very efficient method on massively parallel computers. However, it has never been used since OPA 8.0. The current version in *NEMO* should not even successfully go through the compilation phase.

The surface pressure gradient is computed in *dynspg.F90*. The solver used (PCG or SOR) depending on the value of *nsolv* (namelist parameter). At each time step the time derivative of the barotropic streamfunction is the solution of (2.29). Introducing the following coefficients :

$$
\begin{aligned}
C_{i,j}^{NS} &= \frac{e_{2v}(i,j)}{(H_v(i,j)e_{1v}(i,j))} \\
C_{i,j}^{EW} &= \frac{e_{1u}(i,j)}{(H_u(i,j)e_{2u}(i,j))} \\
B_{i,j} &= \delta_i\left(e_{2v}M_v\right) - \delta_j\left(e_{1u}M_u\right)
\end{aligned}
\qquad (10.2)
$$

the five-point finite difference equation (2.29) can be rewritten as :

$$C^{NS}_{i+1,j}\left(\frac{\partial\psi}{\partial t}\right)_{i+1,j} + C^{EW}_{i,j+1}\left(\frac{\partial\psi}{\partial t}\right)_{i,j+1} + C^{NS}_{i,j}\left(\frac{\partial\psi}{\partial t}\right)_{i-1,j} + C^{EW}_{i,j}\left(\frac{\partial\psi}{\partial t}\right)_{i,j-1}$$
$$- \left(C^{NS}_{i+1,j} + C^{EW}_{i,j+1} + C^{NS}_{i,j} + C^{EW}_{i,j}\right)\left(\frac{\partial\psi}{\partial t}\right)_{i,j} = B_{i,j} \quad (10.3)$$

(10.3) is a linear symmetric system of equations. All the elements of the corresponding matrix **A** vanish except those of five diagonals. With the natural ordering of the grid points (i.e. from west to east and from south to north), the structure of **A** is block-tridiagonal with tridiagonal or diagonal blocks. **A** is a positive-definite symmetric matrix of size $(jpi \cdot jpj)^2$, and **B**, the right hand side of (10.3), is a vector.

10.7.1 Successive Over Relaxation *nsolv*=2

Let us introduce the four cardinal coefficients : $A^S_{i,j} = C^{NS}_{i,j}/D_{i,j}$, $A^W_{i,j} = C^{EW}_{i,j}/D_{i,j}$, $A^E_{i,j} = C^{EW}_{i,j+1}/D_{i,j}$ and $A^N_{i,j} = C^{NS}_{i+1,j}/D_{i,j}$, and define $\tilde{B}_{i,j} = B_{i,j}/D_{i,j}$, where $D_{i,j} = C^{NS}_{i,j} + C^{NS}_{i+1,j} + C^{EW}_{i,j} + C^{EW}_{i,j+1}$ (i.e. the diagonal of **A**). (VII.5.1) can be rewritten as :

$$A^N_{i,j}\left(\frac{\partial\psi}{\partial t}\right)_{i+1,j} + A^E_{i,j}\left(\frac{\partial\psi}{\partial t}\right)_{i,j+1}$$
$$+ A^S_{i,j}\left(\frac{\partial\psi}{\partial t}\right)_{i-1,j} + A^W_{i,j}\left(\frac{\partial\psi}{\partial t}\right)_{i,j-1} - \left(\frac{\partial\psi}{\partial t}\right)_{i,j} = \tilde{B}_{i,j} \quad (10.4)$$

The SOR method used is an iterative method. Its algorithm can be summarised as follows [see **?** for a further discussion] :

initialisation (evaluate a first guess from previous time step computations)

$$\left(\frac{\partial\psi}{\partial t}\right)^0_{i,j} = 2\left(\frac{\partial\psi}{\partial t}\right)^t_{i,j} - \left(\frac{\partial\psi}{\partial t}\right)^{t-1}_{i,j} \quad (10.5)$$

iteration n, from $n = 0$ until convergence, do :

$$R^n_{i,j} = A^N_{i,j}\left(\frac{\partial\psi}{\partial t}\right)^n_{i+1,j} + A^E_{i,j}\left(\frac{\partial\psi}{\partial t}\right)^n_{i,j+1}$$
$$+ A^S_{i,j}\left(\frac{\partial\psi}{\partial t}\right)^{n+1}_{i-1,j} + A^W_{i,j}\left(\frac{\partial\psi}{\partial t}\right)^{n+1}_{i,j-1} - \left(\frac{\partial\psi}{\partial t}\right)^n_{i,j} - \tilde{B}_{i,j} \quad (10.6)$$

$$\left(\frac{\partial\psi}{\partial t}\right)^{n+1}_{i,j} = \left(\frac{\partial\psi}{\partial t}\right)^n_{i,j} + \omega\, R^n_{i,j} \quad (10.7)$$

where ω satisfies $1 \leq \omega \leq 2$. An optimal value exists for ω which significantly accelerates the convergence, but it has to be adjusted empirically for each model domain (except for a uniform grid where an analytical expression for ω can be found [**?**]). The value of ω is set using *sor*, a **namelist** parameter. The convergence test is of the form :

$$\delta = \frac{\sum\limits_{i,j} R^n_{i,j} R^n_{i,j}}{\sum\limits_{i,j} \tilde{B}^n_{i,j} \tilde{B}^n_{i,j}} \leq \epsilon \tag{10.8}$$

where ϵ is the absolute precision that is required. It is recommended that a value smaller or equal to 10^{-3} is used for ϵ since larger values may lead to numerically induced basin scale barotropic oscillations. In fact, for an eddy resolving configuration or in a filtered free surface case, a value three orders of magnitude smaller than this should be used. The precision is specified by setting *eps* (**namelist parameter**). In addition, two other tests are used to halt the iterative algorithm. They involve the number of iterations and the modulus of the right hand side. If the former exceeds a specified value, *nmax* (**namelist parameter**), or the latter is greater than 10^{15}, the whole model computation is stopped and the last computed time step fields are saved in the standard output file. In both cases, this usually indicates that there is something wrong in the model configuration (an error in the mesh, the initial state, the input forcing, or the magnitude of the time step or of the mixing coefficients). A typical value of $nmax$ is a few hundred when $\epsilon = 10^{-6}$, increasing to a few thousand when $\epsilon = 10^{-12}$. The vectorization of the SOR algorithm is not straightforward. The scheme contains two linear recurrences on i and j. This inhibits the vectorisation. Therefore it has been rewritten as :

$$\begin{aligned} R^n_{i,j} = A^N_{i,j} \left(\frac{\partial \psi}{\partial t} \right)^n_{i+1,j} + A^E_{i,j} \left(\frac{\partial \psi}{\partial t} \right)^n_{i,j+1} \\ + A^S_{i,j} \left(\frac{\partial \psi}{\partial t} \right)^n_{i-1,j} + A^W_{i,j} \left(\frac{\partial \psi}{\partial t} \right)^n_{i,j-1} - \left(\frac{\partial \psi}{\partial t} \right)^n_{i,j} - \tilde{B}_{i,j} \end{aligned} \tag{10.9}$$

$$R^n_{i,j} = R^n_{i,j} - \omega \; A^S_{i,j} \; R^n_{i,j-1} \tag{10.10}$$

$$R^n_{i,j} = R^n_{i,j} - \omega \; A^W_{i,j} \; R^n_{i-1,j} \tag{10.11}$$

The SOR method is very flexible and can be used under a wide range of conditions, including irregular boundaries, interior boundary points, etc. Proofs of convergence, etc. may be found in the standard numerical methods texts for partial differential equations.

10.7.2 Preconditioned Conjugate Gradient

(*nbsfs=1*, **namelist parameter**)

A is a definite positive symmetric matrix, thus solving the linear system (10.3) is equivalent to the minimisation of a quadratic functional :

$$\mathbf{Ax} = \mathbf{b} \leftrightarrow \mathbf{x} = \inf_y \phi(\mathbf{y}) \quad , \qquad \phi(\mathbf{y}) = 1/2\langle \mathbf{Ay}, \mathbf{y}\rangle - \langle \mathbf{b}, \mathbf{y}\rangle$$

where $\langle , \rangle$ is the canonical dot product. The idea of the conjugate gradient method is to search for the solution in the following iterative way : assuming that $\mathbf{x}^n$ has been obtained, $\mathbf{x}^{n+1}$ is found from $\mathbf{x}^{n+1} = \mathbf{x}^n + \alpha^n \mathbf{d}^n$ which satisfies :

$$\mathbf{x}^{n+1} = \inf_{\mathbf{y}=\mathbf{x}^n+\alpha^n \ \mathbf{d}^n} \phi(\mathbf{y}) \quad \Leftrightarrow \quad \frac{d\phi}{d\alpha} = 0$$

and expressing $\phi(\mathbf{y})$ as a function of α, we obtain the value that minimises the functional :

$$\alpha^n = \langle \mathbf{r}^n, \mathbf{r}^n\rangle / \langle \mathbf{A}\,\mathbf{d}^n, \mathbf{d}^n\rangle$$

where $\mathbf{r}^n = \mathbf{b} - \mathbf{A}\,\mathbf{x}^n = \mathbf{A}(\mathbf{x} - \mathbf{x}^n)$ is the error at rank n. The descent vector $\mathbf{d}^n$ s chosen to be dependent on the error : $\mathbf{d}^n = \mathbf{r}^n + \beta^n\,\mathbf{d}^{n-1}$. β^n is searched such that the descent vectors form an orthogonal basis for the dot product linked to **A**. Expressing the condition $\langle \mathbf{A}\,\mathbf{d}^n, \mathbf{d}^{n-1}\rangle = 0$ the value of β^n is found : $\beta^n = \langle \mathbf{r}^n, \mathbf{r}^n\rangle / \langle \mathbf{r}^{n-1}, \mathbf{r}^{n-1}\rangle$. As a result, the errors $\mathbf{r}^n$ form an orthogonal base for the canonic dot product while the descent vectors $\mathbf{d}^n$ form an orthogonal base for the dot product linked to **A**. The resulting algorithm is thus the following one :

initialisation :

$$\mathbf{x}^0 = \left(\frac{\partial\psi}{\partial t}\right)^0_{i,j} = 2\left(\frac{\partial\psi}{\partial t}\right)^t_{i,j} - \left(\frac{\partial\psi}{\partial t}\right)^{t-1}_{i,j} \text{, the initial guess}$$

$$\mathbf{r}^0 = \mathbf{d}^0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}^0$$

$$\gamma_0 = \langle \mathbf{r}^0, \mathbf{r}^0\rangle$$

iteration n, from $n = 0$ until convergence, do :

$$\begin{aligned} \mathrm{z}^n &= \mathbf{A}\,\mathbf{d}^n \\ \alpha_n &= \gamma_n \langle \mathbf{z}^n, \mathbf{d}^n\rangle \\ \mathbf{x}^{n+1} &= \mathbf{x}^n + \alpha_n\,\mathbf{d}^n \\ \mathbf{r}^{n+1} &= \mathbf{r}^n - \alpha_n\,\mathbf{z}^n \\ \gamma_{n+1} &= \langle \mathbf{r}^{n+1}, \mathbf{r}^{n+1}\rangle \\ \beta_{n+1} &= \gamma_{n+1}/\gamma_n \\ \mathbf{d}^{n+1} &= \mathbf{r}^{n+1} + \beta_{n+1}\,\mathbf{d}^n \end{aligned} \tag{10.12}$$

The convergence test is :

$$\delta = \gamma_n \,/ \langle \mathbf{b}, \mathbf{b}\rangle \leq \epsilon \tag{10.13}$$

where ϵ is the absolute precision that is required. As for the SOR algorithm, the whole model computation is stopped when the number of iterations, $nmax$, or the modulus of the right hand side of the convergence equation exceeds a specified value (see §10.7.1 for a further discussion). The required precision and the maximum number of iterations allowed are specified by setting eps and $nmax$ (**namelist parameters**).

It can be demonstrated that the above algorithm is optimal, provides the exact solution in a number of iterations equal to the size of the matrix, and that the convergence rate is faster as the matrix is closer to the identity matrix, $i.e.$ its eigenvalues are closer to 1. Therefore, it is more efficient to solve a better conditioned system which has the same solution. For that purpose, we introduce a preconditioning matrix **Q** which is an approximation of **A** but much easier to invert than **A**, and solve the system :

$$\mathbf{Q}^{-1}\mathbf{A}\,\mathbf{x} = \mathbf{Q}^{-1}\mathbf{b} \tag{10.14}$$

The same algorithm can be used to solve (10.14) if instead of the canonical dot product the following one is used : $\langle \mathbf{a}, \mathbf{b} \rangle_Q = \langle \mathbf{a}, \mathbf{Q}\,\mathbf{b} \rangle$, and if $\tilde{\mathbf{b}} = \mathbf{Q}^{-1}\ \mathbf{b}$ and $\tilde{\mathbf{A}} = \mathbf{Q}^{-1}\ \mathbf{A}$ are substituted to **b** and **A** [**?**]. In *NEMO* , **Q** is chosen as the diagonal of **A**, i.e. the simplest form for **Q** so that it can be easily inverted. In this case, the discrete formulation of (10.14) is in fact given by (10.4) and thus the matrix and right hand side are computed independently from the solver used.

10.7.3 FETI

FETI is a powerful solver that was developed by Marc Guyon [**??**]. It has been converted from Fortan 77 to 90, but never successfully tested after that.

Its main advantage is to save a lot of CPU time when compared to the SOR or PCG algorithms. However, its main drawback is that the solution is dependent on the domain decomposition chosen. Using a different number of processors, the solution is the same at the precision required, but not the same at the computer precision. This make it hard to debug.

10.7.4 Boundary Conditions — Islands (key_islands)

The boundary condition used for both recommended solvers is that the time derivative of the barotropic streamfunction is zero along all the coastlines. When islands are present in the model domain, additional computations must be performed to determine the barotropic streamfunction with the correct boundary conditions. This is detailed below.

The model does not have specialised code for islands. They must instead be identified to the solvers by the user via bathymetry information, i.e. the $mbathy$ array should equal -1 over the first island, -2 over the second, ... , $-N$ over the N^{th} island. The model determines the position of island grid-points and defines a closed contour around each island which is used to compute the circulation around each one. The closed contour is formed from the ocean grid-points which are the closest to the island.

First, the island barotropic streamfunctions ψ_n are computed using the SOR or PCG scheme (they are solutions of (10.3) with the right-hand side equal to zero and with $\psi_n = 1$ along the island n and $\psi_n = 0$ along the other coastlines) (Note that specifying 1 as boundary condition on an island for ψ is equivalent to defining a specific right hand side for (10.3)). The requested precision of this computation can be very high since it is only performed once. The absolute precision, $epsisl$, and the maximum number of iterations allowed, $nmisl$, are the **namelist parameters** used for this computation. Their typical values are $epsisl = 10^{-10}$ and $nmisl = 4000$. Then the island matrix A is computed from (2.16) and inverted. At each time step, ψ_0, the solution of (10.3) with $\psi = 0$ along all coastlines, is computed using either SOR or PCG. (It should be noted that the first guess of this computation remains as usual except that $\partial_t \psi_0$ is used, instead of $\partial_t \psi$. Indeed we are computing $\partial_t \psi_0$ which is usually very different from $\partial_t \psi$.) Then, it is easy to find the time evolution of the barotropic streamfunction on each island and to deduce $\partial_t \psi$, and to use it to compute the surface pressure gradient. Note that the value of the barotropic streamfunction itself is also computed as the time integration of $\partial_t \psi$ for further diagnostics.

10.8 Diagnostics

10.8.1 Standard Model Output (default option or key_dimg)

The model outputs are of three types : the restart file, the output listing, and the output file(s). The restart file is used internally by the code when the user wants to start the model with initial conditions defined by a previous simulation. It contains all the information that is necessary in order for there to be no changes in the model results (even at the computer precision) between a run performed with several restarts and the same run performed in one step. It should be noted that this requires that the restart file contain two consecutive time steps for all the prognostic variables, and that it is saved in the same binary format as the one used by the computer that is to read it (in particular, 32 bits binary IEEE format must not be used for this file). The output listing and file(s) are predefined but should be checked and eventually adapted to the user's needs. The output listing is stored in the $ocean.output$ file. The information is printed from within the code on the logical unit $numout$. To locate these prints, use the UNIX command "$grep - inumout^*$" in the source code directory.

In the standard configuration, the user will find the model results in two output files containing values for every time-step where output is demanded : a **VO** file containing all the three dimensional fields in logical unit $numwvo$, and a **SO** file containing all the two dimensional (horizontal) fields in logical unit $numwso$. These outputs are defined in the $diawri.F$ routine. The default and user-selectable diagnostics are described in §III-12-c.

The default output (for all output files apart from the restart file) is 32 bits binary IEEE format, compatible with the Vairmer software (see the Climate Modelling and global Change team WEB server at CERFACS : http ://www.cerfacs.fr). The model's reference directory also contains a visualisation tool based on **NCAR Graphics** (http ://ngwww.ucar.edu).

If a user has access to the NCAR software, he or she can copy the **LODMODEL/UTILS/OPADRA** directory from the reference and, following the **README**, create graphical outputs from the model's results.

10.8.2 Tracer/Dynamics Trends (key_trdlmd, key_diatrdtra, key_diatrddyn)

When **key_diatrddyn** and/or **key_diatrddyn** cpp variables are defined, each trend of the dynamics and/or temperature and salinity time evolution equations is stored in three-dimensional arrays just after their computation ($i.e.$ at the end of each $dyn \cdots .F90$ and/or $tra \cdots .F90$ routine). These trends are then used in diagnostic routines $diadyn.F90$ and $diatra.F90$ respectively. In the standard model, these routines check the basin averaged properties of the momentum and tracer equations every *ntrd* time-steps (**namelist parameter**). These routines are supplied as an example ; they must be adapted by the user to his/her requirements.

These two options imply the creation of several extra arrays in the in-core memory, increasing quite seriously the code memory requirements.

10.8.3 On-line Floats trajectories

```
!-----------------------------------------------------------------------
&namflo       !   float parameters                                      ("key_float")
!-----------------------------------------------------------------------
    ln_rstflo = .false.    !  float restart (T) or not (F)
    nwritefl  =      75    !  frequency of writing in float output file
    nstockfl  =    5475    !  frequency of creation of the float restart file
    ln_argo   = .false.    !  Argo type floats (stay at the surface each 10 days)
    ln_flork4 = .false.    !  trajectories computed with a 4th order Runge-Kutta (T)
                           !  or computed with Blanke' scheme (F)
/
```

The on-line computation of floats adevected either by the three dimensional velocity field or constraint to remain at a given depth ($w = 0$ in the computation) have been introduced in the system during the CLIPPER project. The algorithm used is based on the work of **?**. (see also the web site describing the off-line use of this marvellous diagnostic tool (http ://stockage.univ-brest.fr/ grima/Ariane/).

10.8.4 Other Diagnostics

Aside from the standard model variables, other diagnostics can be computed on-line or can be added to the model. The available ready-to-add diagnostics routines can be found in directory DIA. Among the available diagnostics are :

- the mixed layer depth (based on a density criterion) (*diamxl.F90*)
- the turbocline depth (based on a turbulent mixing coefficient criterion) (*diamxl.F90*)
- the depth of the 20 ˚ C isotherm (*diahth.F90*)
- the depth of the thermocline (maximum of the vertical temperature gradient) (*diahth.F90*)
- the meridional heat and salt transports and their decomposition (*diamfl.F90*)
- the surface pressure (*diaspr.F90*)

Curvilinear s-Coordinate Equations

Contents

In order to establish the set of Primitive Equation in curvilinear s-coordinates ($i.e.$ an orthogonal curvilinear coordinate in the horizontal and s-coordinate in the vertical), we start from the set of equations established in §2.3.2 for the special case $k = z$ and thus $e_3 = 1$, and we introduce an arbitrary vertical coordinate $s = s(i, j, z, t)$. Let us define a new vertical scale factor by $e_3 = \partial z/\partial s$ (which now depends on (i, j, z, t)) and the horizontal slope of s-surfaces by :

$$\sigma_1 = \frac{1}{e_1} \left.\frac{\partial z}{\partial i}\right|_s \quad \text{and} \quad \sigma_2 = \frac{1}{e_2} \left.\frac{\partial z}{\partial j}\right|_s \tag{A.1}$$

The chain rule to establish the model equations in the curvilinear s-coordinate system is :

$$\begin{aligned}
\left.\frac{\partial \bullet}{\partial t}\right|_z &= \left.\frac{\partial \bullet}{\partial t}\right|_s - \frac{\partial \bullet}{\partial s}\frac{\partial s}{\partial t} \\
\left.\frac{\partial \bullet}{\partial i}\right|_z &= \left.\frac{\partial \bullet}{\partial i}\right|_s - \frac{\partial \bullet}{\partial s}\frac{\partial s}{\partial i} = \left.\frac{\partial \bullet}{\partial i}\right|_s - \frac{e_1}{e_3}\sigma_1 \frac{\partial \bullet}{\partial s} \\
\left.\frac{\partial \bullet}{\partial j}\right|_z &= \left.\frac{\partial \bullet}{\partial j}\right|_s - \frac{\partial \bullet}{\partial s}\frac{\partial s}{\partial j} = \left.\frac{\partial \bullet}{\partial j}\right|_s - \frac{e_2}{e_3}\sigma_2 \frac{\partial \bullet}{\partial s} \\
\frac{\partial \bullet}{\partial z} &= \frac{1}{e_3}\frac{\partial \bullet}{\partial s}
\end{aligned} \tag{A.2}$$

In particular applying the time derivative chain rule to z provides the expression for w_s, the vertical velocity of the $s-$surfaces :

$$w_s = \left.\frac{\partial z}{\partial t}\right|_s = \frac{\partial z}{\partial s}\,\frac{\partial s}{\partial t} = e_3\,\frac{\partial s}{\partial t} \tag{A.3}$$

A.1 Continuity Equation

Using (A.2) and the fact that the horizontal scale factors e_1 and e_2 do not depend on the vertical coordinate, the divergence of the velocity relative to the (i,j,z) coordinate system is transformed as follows :

$$\begin{aligned}
\nabla\cdot\mathbf{U} &= \frac{1}{e_1\,e_2}\left[\left.\frac{\partial(e_2\,u)}{\partial i}\right|_z + \left.\frac{\partial(e_1\,v)}{\partial j}\right|_z\right] + \frac{\partial w}{\partial z} \\
&= \frac{1}{e_1\,e_2}\left[\left.\frac{\partial(e_2\,u)}{\partial i}\right|_s - \frac{e_1}{e_3}\sigma_1\frac{\partial(e_2\,u)}{\partial s} + \left.\frac{\partial(e_1\,v)}{\partial j}\right|_s - \frac{e_2}{e_3}\sigma_2\frac{\partial(e_1\,v)}{\partial s}\right] + \frac{\partial w}{\partial s}\,\frac{\partial s}{\partial z} \\
&= \frac{1}{e_1\,e_2}\left[\left.\frac{\partial(e_2\,u)}{\partial i}\right|_s + \left.\frac{\partial(e_1\,v)}{\partial j}\right|_s\right] + \frac{1}{e_3}\left[\frac{\partial w}{\partial s} - \sigma_1\frac{\partial u}{\partial s} - \sigma_2\frac{\partial v}{\partial s}\right] \\
&= \frac{1}{e_1\,e_2\,e_3}\left[\left.\frac{\partial(e_2\,e_3\,u)}{\partial i}\right|_s - e_2\,u\left.\frac{\partial e_3}{\partial i}\right|_s + \left.\frac{\partial(e_1\,e_3\,v)}{\partial j}\right|_s - e_1 v\left.\frac{\partial e_3}{\partial j}\right|_s\right] \\
&\qquad + \frac{1}{e_3}\left[\frac{\partial w}{\partial s} - \sigma_1\frac{\partial u}{\partial s} - \sigma_2\frac{\partial v}{\partial s}\right]
\end{aligned}$$

Noting that $\frac{1}{e_1}\left.\frac{\partial e_3}{\partial i}\right|_s = \frac{1}{e_1}\left.\frac{\partial^2 z}{\partial i\,\partial s}\right|_s = \frac{\partial}{\partial s}\left(\frac{1}{e_1}\left.\frac{\partial z}{\partial i}\right|_s\right) = \frac{\partial \sigma_1}{\partial s}$ and $\frac{1}{e_2}\left.\frac{\partial e_3}{\partial j}\right|_s = \frac{\partial \sigma_2}{\partial s}$, it becomes :

$$\begin{aligned}
\nabla\cdot\mathbf{U} &= \frac{1}{e_1\,e_2\,e_3}\left[\left.\frac{\partial(e_2\,e_3\,u)}{\partial i}\right|_s + \left.\frac{\partial(e_1\,e_3\,v)}{\partial j}\right|_s\right] \\
&\qquad + \frac{1}{e_3}\left[\frac{\partial w}{\partial s} - u\frac{\partial \sigma_1}{\partial s} - v\frac{\partial \sigma_2}{\partial s} - \sigma_1\frac{\partial u}{\partial s} - \sigma_2\frac{\partial v}{\partial s}\right] \\
&= \frac{1}{e_1\,e_2\,e_3}\left[\left.\frac{\partial(e_2\,e_3\,u)}{\partial i}\right|_s + \left.\frac{\partial(e_1\,e_3\,v)}{\partial j}\right|_s\right] + \frac{1}{e_3}\,\frac{\partial}{\partial s}\left[w - u\,\sigma_1 - v\,\sigma_2\right]
\end{aligned}$$

Here, w is the vertical velocity relative to the $z-$coordinate system. Introducing the dia-surface velocity component, ω, defined as the velocity relative to the moving s-surfaces and normal to them :

$$\omega = w - w_s - \sigma_1\, u - \sigma_2\, v \tag{A.4}$$

with w_s given by (A.3), we obtain the expression for the divergence of the velocity in the curvilinear s-coordinate system :

$$\begin{aligned}
\nabla\cdot\mathbf{U} &= \frac{1}{e_1\,e_2\,e_3}\left[\left.\frac{\partial(e_2\,e_3\,u)}{\partial i}\right|_s + \left.\frac{\partial(e_1\,e_3\,v)}{\partial j}\right|_s\right] + \frac{1}{e_3}\frac{\partial\omega}{\partial s} + \frac{1}{e_3}\frac{\partial w_s}{\partial s} \\
&= \frac{1}{e_1\,e_2\,e_3}\left[\left.\frac{\partial(e_2\,e_3\,u)}{\partial i}\right|_s + \left.\frac{\partial(e_1\,e_3\,v)}{\partial j}\right|_s\right] + \frac{1}{e_3}\frac{\partial\omega}{\partial s} + \frac{1}{e_3}\frac{\partial}{\partial s}\left(e_3\,\frac{\partial s}{\partial t}\right) \\
&= \frac{1}{e_1\,e_2\,e_3}\left[\left.\frac{\partial(e_2\,e_3\,u)}{\partial i}\right|_s + \left.\frac{\partial(e_1\,e_3\,v)}{\partial j}\right|_s\right] + \frac{1}{e_3}\frac{\partial\omega}{\partial s} + \frac{\partial}{\partial s}\frac{\partial s}{\partial t} + \frac{1}{e_3}\frac{\partial s}{\partial t}\frac{\partial e_3}{\partial s} \\
&= \frac{1}{e_1\,e_2\,e_3}\left[\left.\frac{\partial(e_2\,e_3\,u)}{\partial i}\right|_s + \left.\frac{\partial(e_1\,e_3\,v)}{\partial j}\right|_s\right] + \frac{1}{e_3}\frac{\partial\omega}{\partial s} + \frac{1}{e_3}\frac{\partial e_3}{\partial t}
\end{aligned}$$

As a result, the continuity equation (2.1c) in the s-coordinates becomes :

$$\frac{1}{e_3}\frac{\partial e_3}{\partial t} + \frac{1}{e_1\,e_2\,e_3}\left[\left.\frac{\partial(e_2\,e_3\,u)}{\partial i}\right|_s + \left.\frac{\partial(e_1\,e_3\,v)}{\partial j}\right|_s\right] + \frac{1}{e_3}\frac{\partial\omega}{\partial s} = 0 \tag{A.5}$$

A.2 Momentum Equation

Let us consider (2.25), the first component of the momentum equation in the vector invariant form (similar manipulations can be performed on the second component). Its non-linear term can be transformed as follows :

$$+\ \zeta\big|_z\, v - \frac{1}{2e_1}\ \frac{\partial(u^2+v^2)}{\partial i}\bigg|_z - w\ \frac{\partial u}{\partial z}$$

$$= \frac{1}{e_1\, e_2}\left[\frac{\partial(e_2\, v)}{\partial i}\bigg|_z - \frac{\partial(e_1\, u)}{\partial j}\bigg|_z\right]\ v - \frac{1}{2e_1}\ \frac{\partial(u^2+v^2)}{\partial i}\bigg|_z - w\frac{\partial u}{\partial z}$$

$$= \frac{1}{e_1\, e_2}\left[\frac{\partial(e_2\, v)}{\partial i}\bigg|_s - \frac{\partial(e_1\, u)}{\partial j}\bigg|_s - \frac{e_1}{e_3}\sigma_1\frac{\partial(e_2\, v)}{\partial s} + \frac{e_2}{e_3}\sigma_2\frac{\partial(e_1\, u)}{\partial s}\right]\ v$$
$$-\frac{1}{2e_1}\left(\frac{\partial(u^2+v^2)}{\partial i}\bigg|_s - \frac{e_1}{e_3}\sigma_1\frac{\partial(u^2+v^2)}{\partial s}\right) - \frac{w}{e_3}\frac{\partial u}{\partial s}$$

$$= \zeta\big|_s\ v - \frac{1}{2\, e_1}\ \frac{\partial(u^2+v^2)}{\partial i}\bigg|_s - \frac{w}{e_3}\frac{\partial u}{\partial s} - \left[\frac{\sigma_1}{e_3}\frac{\partial v}{\partial s} - \frac{\sigma_2}{e_3}\frac{\partial u}{\partial s}\right]\ v$$
$$+\ \frac{\sigma_1}{2e_3}\frac{\partial(u^2+v^2)}{\partial s}$$

$$= \zeta\big|_s\ v - \frac{1}{2e_1}\ \frac{\partial(u^2+v^2)}{\partial i}\bigg|_s$$
$$-\ \frac{1}{e_3}\left[w\frac{\partial u}{\partial s} + \sigma_1 v\frac{\partial v}{\partial s} - \sigma_2 v\frac{\partial u}{\partial s} - \sigma_1 u\frac{\partial u}{\partial s} - \sigma_1 v\frac{\partial v}{\partial s}\right]$$

$$= \zeta\big|_s\ v - \frac{1}{2e_1}\ \frac{\partial(u^2+v^2)}{\partial i}\bigg|_s - \frac{1}{e_3}\left[w - \sigma_2 v - \sigma_1 u\right]\ \frac{\partial u}{\partial s}$$

$$= \zeta\big|_s\ v - \frac{1}{2e_1}\ \frac{\partial(u^2+v^2)}{\partial i}\bigg|_s - \frac{1}{e_3}\omega\frac{\partial u}{\partial s} - \frac{\partial s}{\partial t}\frac{\partial u}{\partial s}$$

Therefore, the non-linear terms of the momentum equation have the same form in $z-$ and $s-$coordinates but with the addition of the time derivative of the velocity :

$$+\ \zeta\big|_z\, v - \frac{1}{2e_1}\ \frac{\partial(u^2+v^2)}{\partial i}\bigg|_z - w\ \frac{\partial u}{\partial z}$$
$$= -\frac{\partial u}{\partial t} + \zeta\big|_s\ v - \frac{1}{2e_1}\ \frac{\partial(u^2+v^2)}{\partial i}\bigg|_s - \frac{1}{e_3}\omega\frac{\partial u}{\partial s} \qquad \text{(A.6)}$$

The pressure gradient term can be transformed as follows :

$$
\begin{aligned}
-\frac{1}{\rho_o e_1} \left.\frac{\partial p}{\partial i}\right|_z &= -\frac{1}{\rho_o e_1}\left[\left.\frac{\partial p}{\partial i}\right|_s - \frac{e_1}{e_3}\sigma_1 \frac{\partial p}{\partial s}\right] \\
&= -\frac{1}{\rho_o\, e_1} \left.\frac{\partial p}{\partial i}\right|_s + \frac{\sigma_1}{\rho_o\, e_3}\left(-g\;\rho\; e_3\right) \qquad \text{(A.7)} \\
&= -\frac{1}{\rho_o\, e_1} \left.\frac{\partial p}{\partial i}\right|_s - \frac{g\;\rho}{\rho_o}\sigma_1
\end{aligned}
$$

An additional term appears in (A.7) which accounts for the tilt of model levels.

Introducing (A.6) and (A.7) in (2.25) and regrouping the time derivative terms in the left hand side, and performing the same manipulation on the second component, we obtain the vector invariant form of the momentum equations in the $s-$coordinate :

$$
\begin{aligned}
\frac{1}{e_3}\frac{\partial\left(e_3\, u\right)}{\partial t} = &+\left(\zeta+f\right)\, v - \frac{1}{2\, e_1}\frac{\partial}{\partial i}\left(u^2+v^2\right) - \frac{1}{e_3}\omega\frac{\partial u}{\partial k} \\
&- \frac{1}{e_1}\frac{\partial}{\partial i}\left(\frac{p_s+p_h}{\rho_o}\right) + g\frac{\rho}{\rho_o}\sigma_1 + D_u^{\mathbf{U}} + F_u^{\mathbf{U}} \qquad \text{(A.8a)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{1}{e_3}\frac{\partial\left(e_3\, v\right)}{\partial t} = &-\left(\zeta+f\right)\, u - \frac{1}{2\, e_2}\frac{\partial}{\partial j}\left(u^2+v^2\right) - \frac{1}{e_3}\omega\frac{\partial v}{\partial k} \\
&- \frac{1}{e_2}\frac{\partial}{\partial j}\left(\frac{p_s+p_h}{\rho_o}\right) + g\frac{\rho}{\rho_o}\sigma_2 + D_v^{\mathbf{U}} + F_v^{\mathbf{U}} \qquad \text{(A.8b)}
\end{aligned}
$$

It has the same form as in the $z-$coordinate but for the vertical scale factor that has appeared inside the time derivative. The form of the vertical physics and forcing terms remains unchanged. The form of the lateral physics is discussed in appendix B.

A.3 Tracer Equation

The tracer equation is obtained using the same calculation as for the continuity equation and then regrouping the time derivative terms in the left hand side :

$$
\begin{aligned}
\frac{1}{e_3}\frac{\partial\left(e_3 T\right)}{\partial t} = &-\frac{1}{e_1\, e_2\, e_3}\left[\frac{\partial}{\partial i}\left(e_2\, e_3\; Tu\right) + \frac{\partial}{\partial j}\left(e_1\, e_3\; Tv\right)\right. \\
&\left. + \frac{\partial}{\partial j}\left(e_1\, e_3\; Tv\right)\right] + D^T + F^T \qquad \text{(A.9)}
\end{aligned}
$$

The expression for the advection term is a straight consequence of (A.4), the expression of the 3D divergence in the s-coordinates established above.

Appendix B : Diffusive Operators

Contents

B.1 Horizontal/Vertical 2nd Order Tracer Diffusive Operators

In the z-coordinate, the horizontal/vertical second order tracer diffusion operator is given by :

$$\begin{split} D^T = \frac{1}{e_1\, e_2} \left[\frac{\partial}{\partial i} \left(\frac{e_2}{e_1} A^{lT} \; \left. \frac{\partial T}{\partial i} \right|_z \right) \right|_z \\ + \frac{\partial}{\partial j} \left(\frac{e_1}{e_2} A^{lT} \; \left. \frac{\partial T}{\partial j} \right|_z \right) \left| _z \right] + \frac{\partial}{\partial z} \left(A^{vT} \; \frac{\partial T}{\partial z} \right) \end{split} \tag{B.1}$$

In the s-coordinate, we defined the slopes of s-surfaces, σ_1 and σ_2 by (!!!A.1!!!) and the vertical/horizontal ratio of diffusion coefficient by $\epsilon = A^{vT}/A^{lT}$. The diffusion operator is given by :

$$D^T = \nabla|_s \cdot \left[A^{lT} \; \Re \cdot \nabla|_s \, T \right] \quad \text{where } \Re = \begin{pmatrix} 1 & 0 & -\sigma_1 \\ 0 & 1 & -\sigma_2 \\ -\sigma_1 & -\sigma_2 & \varepsilon + \sigma_1^2 + \sigma_2^2 \end{pmatrix} \tag{B.2}$$

or in expanded form :

$$\begin{aligned}D^T = \frac{1}{e_1\,e_2\,e_3} \Bigg[\quad & e_2\,e_3\,A^{lT}\ \frac{\partial}{\partial i}\left(\frac{1}{e_1}\ \left.\frac{\partial T}{\partial i}\right|_s - \frac{\sigma_1}{e_3}\ \frac{\partial T}{\partial s} \right)\Bigg|_s \\ & + e_1\,e_3\,A^{lT}\ \frac{\partial}{\partial j}\left(\frac{1}{e_2}\ \left.\frac{\partial T}{\partial j}\right|_s - \frac{\sigma_2}{e_3}\ \frac{\partial T}{\partial s} \right)\Bigg|_s \\ & + e_1\,e_2\,A^{lT}\ \frac{\partial}{\partial s}\left(-\frac{\sigma_1}{e_1}\ \left.\frac{\partial T}{\partial i}\right|_s - \frac{\sigma_2}{e_2}\ \left.\frac{\partial T}{\partial j}\right|_s \right. \\ & \qquad\qquad\qquad \left. + \left(\varepsilon + \sigma_1^2 + \sigma_2^2\right)\ \frac{1}{e_3}\ \frac{\partial T}{\partial s} \right) \ \Bigg]\end{aligned} \tag{B.3}$$

Equation (B.2) (or equivalently (B.3)) is obtained from (B.1) without any additional assumption. Indeed, for the special case $k = z$ and thus $e_3 = 1$, we introduce an arbitrary vertical coordinate $s = s(i, j, z)$ as in Appendix A and use (A.1) and (A.2). Since no cross horizontal derivative $\partial_i\partial_j$ appears in (B.1), the (i,z) and (j,z) planes are independent. The derivation can then be demonstrated for the (i,z) $\rightarrow$ (j,s) transformation without any loss of generality :

$$D^T = \frac{1}{e_1\,e_2}\ \frac{\partial}{\partial i}\left(\frac{e_2}{e_1} A^{lT}\ \left.\frac{\partial T}{\partial i}\right|_z \right)\Bigg|_z + \frac{\partial}{\partial z}\left(A^{vT}\ \frac{\partial T}{\partial z} \right)$$

$$\begin{aligned}= \frac{1}{e_1\,e_2} \Bigg[\ & \frac{\partial}{\partial i}\left(\frac{e_2}{e_1} A^{lT}\ \left(\left.\frac{\partial T}{\partial i}\right|_s - \frac{e_1\,\sigma_1}{e_3}\frac{\partial T}{\partial s} \right)\right)\Bigg|_s \\ & - \frac{e_1\,\sigma_1}{e_3}\frac{\partial}{\partial s}\left(\frac{e_2}{e_1} A^{lT}\ \left(\left.\frac{\partial T}{\partial i}\right|_s - \frac{e_1\,\sigma_1}{e_3}\frac{\partial T}{\partial s} \right)\Bigg|_s \right) \Bigg] + \frac{1}{e_3}\frac{\partial}{\partial s}\left[\frac{A^{vT}}{e_3}\ \frac{\partial T}{\partial s} \right]\end{aligned}$$

$$\begin{aligned}= \frac{1}{e_1\,e_2\,e_3} \Bigg[\ & \frac{\partial}{\partial i}\left(\frac{e_2\,e_3}{e_1} A^{lT}\ \left.\frac{\partial T}{\partial i}\right|_s \right)\Bigg|_s - \frac{e_2}{e_1} A^{lT}\ \left.\frac{\partial e_3}{\partial i}\right|_s\ \left.\frac{\partial T}{\partial i}\right|_s \\ & - e_3\frac{\partial}{\partial i}\left(\frac{e_2\,\sigma_1}{e_3} A^{lT}\ \frac{\partial T}{\partial s} \right)\Bigg|_s - e_1\,\sigma_1\frac{\partial}{\partial s}\left(\frac{e_2}{e_1} A^{lT}\ \left.\frac{\partial T}{\partial i}\right|_s \right) \\ & - e_1\,\sigma_1\frac{\partial}{\partial s}\left(-\frac{e_2\,\sigma_1}{e_3} A^{lT}\ \frac{\partial T}{\partial s} \right) + \frac{\partial}{\partial s}\left(\frac{e_1\,e_2}{e_3} A^{vT}\ \frac{\partial T}{\partial s} \right) \ \Bigg]\end{aligned}$$

Noting that $\frac{1}{e_1}\ \left.\frac{\partial e_3}{\partial i}\right|_s = \frac{\partial \sigma_1}{\partial s}$, it becomes :

$$
\begin{aligned}
= \frac{1}{e_1\,e_2\,e_3} \Bigg[\quad & \frac{\partial}{\partial i}\left(\frac{e_2\,e_3}{e_1} A^{lT} \left. \frac{\partial T}{\partial i} \right|_s \right)\Bigg|_s - e_3 \frac{\partial}{\partial i}\left(\frac{e_2\,\sigma_1}{e_3} A^{lT} \frac{\partial T}{\partial s} \right)\Bigg|_s \\
& - e_2 A^{lT} \frac{\partial \sigma_1}{\partial s} \left. \frac{\partial T}{\partial i} \right|_s - e_1\,\sigma_1 \frac{\partial}{\partial s}\left(\frac{e_2}{e_1} A^{lT} \left. \frac{\partial T}{\partial i} \right|_s \right) \\
& + e_1\,\sigma_1 \frac{\partial}{\partial s}\left(\frac{e_2\,\sigma_1}{e_3} A^{lT} \frac{\partial T}{\partial s} \right) + \frac{\partial}{\partial s}\left(\frac{e_1\,e_2}{e_3} A^{vT} \frac{\partial T}{\partial z} \right) \quad \Bigg]
\end{aligned}
$$

$$
\begin{aligned}
= \frac{1}{e_1\,e_2\,e_3} \Bigg[\quad & \frac{\partial}{\partial i}\left(\frac{e_2\,e_3}{e_1} A^{lT} \left. \frac{\partial T}{\partial i} \right|_s \right)\Bigg|_s - \frac{\partial}{\partial i}\left(e_2\,\sigma_1 A^{lT} \frac{\partial T}{\partial s} \right)\Bigg|_s \\
& + \frac{e_2\,\sigma_1}{e_3} A^{lT} \frac{\partial T}{\partial s} \left. \frac{\partial e_3}{\partial i} \right|_s - e_2 A^{lT} \frac{\partial \sigma_1}{\partial s} \left. \frac{\partial T}{\partial i} \right|_s \\
& - e_2\,\sigma_1 \frac{\partial}{\partial s}\left(A^{lT} \left. \frac{\partial T}{\partial i} \right|_s \right) + \frac{\partial}{\partial s}\left(\frac{e_1\,e_2\,\sigma_1^2}{e_3} A^{lT} \frac{\partial T}{\partial s} \right) \\
& - \frac{\partial\,(e_1\,e_2\,\sigma_1)}{\partial s} \left(\frac{\sigma_1}{e_3} A^{lT} \frac{\partial T}{\partial s} \right) + \frac{\partial}{\partial s}\left(\frac{e_1\,e_2}{e_3} A^{vT} \frac{\partial T}{\partial s} \right) \quad \Bigg]
\end{aligned}
$$

using the same remark as just above, it becomes :

$$
\begin{aligned}
= \frac{1}{e_1\,e_2\,e_3} \Bigg[\quad & \frac{\partial}{\partial i}\left(\frac{e_2\,e_3}{e_1} A^{lT} \left. \frac{\partial T}{\partial i} \right|_s - e_2\,\sigma_1 A^{lT} \frac{\partial T}{\partial s} \right)\Bigg|_s \\
& + \frac{e_1\,e_2\,\sigma_1}{e_3} A^{lT} \frac{\partial T}{\partial s} \frac{\partial \sigma_1}{\partial s} - \frac{\sigma_1}{e_3} A^{lT} \frac{\partial\,(e_1\,e_2\,\sigma_1)}{\partial s} \frac{\partial T}{\partial s} \\
& - e_2 \left(A^{lT} \frac{\partial \sigma_1}{\partial s} \left. \frac{\partial T}{\partial i} \right|_s + \frac{\partial}{\partial s}\left(\sigma_1 A^{lT} \left. \frac{\partial T}{\partial i} \right|_s \right) - \frac{\partial \sigma_1}{\partial s} A^{lT} \left. \frac{\partial T}{\partial i} \right|_s \right) \\
& + \frac{\partial}{\partial s}\left(\frac{e_1\,e_2\,\sigma_1^2}{e_3} A^{lT} \frac{\partial T}{\partial s} + \frac{e_1\,e_2}{e_3} A^{vT} \frac{\partial T}{\partial s} \right) \quad \Bigg]
\end{aligned}
$$

Since the horizontal scale factors do not depend on the vertical coordinate, the last term of the first line and the first term of the last line cancel, while the second line reduces to a single vertical derivative, so it becomes :

$$
\begin{aligned}
= \frac{1}{e_1\,e_2\,e_3} \Bigg[\quad & \frac{\partial}{\partial i}\left(\frac{e_2\,e_3}{e_1} A^{lT} \left. \frac{\partial T}{\partial i} \right|_s - e_2\,\sigma_1\, A^{lT} \frac{\partial T}{\partial s} \right)\Bigg|_s \\
& + \frac{\partial}{\partial s}\left(-e_2\,\sigma_1\, A^{lT} \left. \frac{\partial T}{\partial i} \right|_s + A^{lT} \frac{e_1\,e_2}{e_3} \left(\varepsilon + \sigma_1^2 \right) \frac{\partial T}{\partial s} \right) \quad \Bigg]
\end{aligned}
$$

in other words, the horizontal Laplacian operator in the (i,s) plane takes the following form :

$$D^T = \frac{1}{e_1\,e_2\,e_3} \left(\begin{array}{c} \left.\frac{\partial(e_2 e_3 \bullet)}{\partial i}\right|_s \\ \frac{\partial(e_1 e_2 \bullet)}{\partial s} \end{array} \right) \cdot \left[A^{lT} \left(\begin{array}{cc} 1 & -\sigma_1 \\ -\sigma_1 & \varepsilon_1^2 \end{array} \right) \cdot \left(\begin{array}{c} \frac{1}{e_1} \left.\frac{\partial \bullet}{\partial i}\right|_s \\ \frac{1}{e_3} \frac{\partial \bullet}{\partial s} \end{array} \right) (T) \right]$$

B.2 Iso/diapycnal 2nd Order Tracer Diffusive Operators

The iso/diapycnal diffusive tensor $\mathbf{A_I}$ expressed in the (i,j,k) curvilinear coordinate system in which the equations of the ocean circulation model are formulated, takes the following form [**?**] :

$$\mathbf{A_I} = \frac{A^{lT}}{\left(1 + a_1^2 + a_2^2\right)} \left[\begin{array}{ccc} 1 + a_1^2 & -a_1 a_2 & -a_1 \\ -a_1 a_2 & 1 + a_2^2 & -a_2 \\ -a_1 & -a_2 & \varepsilon + a_1^2 + a_2^2 \end{array} \right]$$

where (a_1, a_2) are the isopycnal slopes in (**i**, **j**) directions :

$$a_1 = \frac{e_3}{e_1} \left(\frac{\partial \rho}{\partial i} \right) \left(\frac{\partial \rho}{\partial k} \right)^{-1} \qquad , \qquad a_2 = \frac{e_3}{e_2} \left(\frac{\partial \rho}{\partial j} \right) \left(\frac{\partial \rho}{\partial k} \right)^{-1}$$

In practice, the isopycnal slopes are generally less than 10^{-2} in the ocean, so $\mathbf{A_I}$ can be simplified appreciably [**?**] :

$$\mathbf{A_I} \approx A^{lT} \left[\begin{array}{ccc} 1 & 0 & -a_1 \\ 0 & 1 & -a_2 \\ -a_1 & -a_2 & \varepsilon + a_1^2 + a_2^2 \end{array} \right]$$

The resulting isopycnal operator conserves the quantity and dissipates its square. The demonstration of the first property is trivial as (B.2) is the divergence of fluxes. Let us demonstrate the second one :

$$\iiint\limits_D T\ \nabla . \left(\mathbf{A_I} \nabla T \right)\ dv = - \iiint\limits_D \nabla T\ . \left(\mathbf{A_I} \nabla T \right)\ dv$$

since

$$\begin{aligned} \nabla T\ . \left(\mathbf{A_I} \nabla T \right) &= A^{lT} \left[\left(\frac{\partial T}{\partial i} \right)^2 - 2 a_1 \frac{\partial T}{\partial i} \frac{\partial T}{\partial k} + \left(\frac{\partial T}{\partial j} \right)^2 \right. \\ &\qquad \left. - 2 a_2 \frac{\partial T}{\partial j} \frac{\partial T}{\partial k} + \left(a_1^2 + a_2^2 \right) \left(\frac{\partial T}{\partial k} \right)^2 \right] \\ &= A_h \left[\left(\frac{\partial T}{\partial i} - a_1 \frac{\partial T}{\partial k} \right)^2 + \left(\frac{\partial T}{\partial j} - a_2 \frac{\partial T}{\partial k} \right)^2 \right] \quad \geq 0 \end{aligned}$$

the property becomes obvious.

The resulting diffusion operator in z-coordinate has the following form :

$$
\begin{aligned}
D^T = \frac{1}{e_1 e_2} \left\{ \frac{\partial}{\partial i} \left[A_h \left(\frac{e_2}{e_1} \frac{\partial T}{\partial i} - a_1 \frac{e_2}{e_3} \frac{\partial T}{\partial k} \right) \right] \right. & \\
\left. + \frac{\partial}{\partial j} \left[A_h \left(\frac{e_1}{e_2} \frac{\partial T}{\partial j} - a_2 \frac{e_1}{e_3} \frac{\partial T}{\partial k} \right) \right] \right\} & \\
+ \frac{1}{e_3} \frac{\partial}{\partial k} \left[A_h \left(-\frac{a_1}{e_1} \frac{\partial T}{\partial i} - \frac{a_2}{e_2} \frac{\partial T}{\partial j} + \frac{\left(a_1^2 + a_2^2\right)}{e_3} \frac{\partial T}{\partial k} \right) \right] &
\end{aligned}
$$

It has to be emphasised that the simplification introduced, leads to a decoupling between (i,z) and (j,z) planes. The operator has therefore the same expression as (B.3), the diffusion operator obtained for geopotential diffusion in the s-coordinate.

B.3 Lateral/Vertical Momentum Diffusive Operators

The second order momentum diffusion operator (Laplacian) in the z-coordinate is found by applying (2.19e), the expression for the Laplacian of a vector, to the horizontal velocity vector :

$$
\Delta \mathbf{U}_h = \nabla \left(\nabla \cdot \mathbf{U}_h \right) - \nabla \times \left(\nabla \times \mathbf{U}_h \right)
$$

$$
= \begin{pmatrix} \frac{1}{e_1} \frac{\partial \chi}{\partial i} \\ \frac{1}{e_2} \frac{\partial \chi}{\partial j} \\ \frac{1}{e_3} \frac{\partial \chi}{\partial k} \end{pmatrix} - \begin{pmatrix} \frac{1}{e_2} \frac{\partial \zeta}{\partial j} - \frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{1}{e_3} \frac{\partial u}{\partial k} \right) \\ \frac{1}{e_3} \frac{\partial}{\partial k} \left(-\frac{1}{e_3} \frac{\partial v}{\partial k} \right) - \frac{1}{e_1} \frac{\partial \zeta}{\partial i} \\ \frac{1}{e_1 e_2} \left[\frac{\partial}{\partial i} \left(\frac{e_2}{e_3} \frac{\partial u}{\partial k} \right) - \frac{\partial}{\partial j} \left(-\frac{e_1}{e_3} \frac{\partial v}{\partial k} \right) \right] \end{pmatrix}
$$

$$
= \begin{pmatrix} \frac{1}{e_1} \frac{\partial \chi}{\partial i} - \frac{1}{e_2} \frac{\partial \zeta}{\partial j} \\ \frac{1}{e_2} \frac{\partial \chi}{\partial j} + \frac{1}{e_1} \frac{\partial \zeta}{\partial i} \\ 0 \end{pmatrix} + \frac{1}{e_3} \begin{pmatrix} \frac{\partial}{\partial k} \left(\frac{1}{e_3} \frac{\partial u}{\partial k} \right) \\ \frac{\partial}{\partial k} \left(\frac{1}{e_3} \frac{\partial v}{\partial k} \right) \\ \frac{\partial \chi}{\partial k} - \frac{1}{e_1 e_2} \left(\frac{\partial^2 (e_2\, u)}{\partial i \partial k} + \frac{\partial^2 (e_1\, v)}{\partial j \partial k} \right) \end{pmatrix}
$$

Using (2.19b), the definition of the horizontal divergence, the third componant of the second vector is obviously zero and thus :

$$
\Delta \mathbf{U}_h = \nabla_h \left(\chi \right) - \nabla_h \times \left(\zeta \right) + \frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{1}{e_3} \frac{\partial\, \mathbf{U}_h}{\partial k} \right)
$$

Note that this operator ensures a full separation between the vorticity and horizontal divergence fields (see Appendix C). It is only equal to a Laplacian applied to each component in Cartesian coordinates, not on the sphere.

The horizontal/vertical second order (Laplacian type) operator used to diffuse horizontal momentum in the z-coordinate therefore takes the following form :

$$\mathbf{D}^{\mathbf{U}} = \nabla_h \left(A^{lm}\; \chi \right) - \nabla_h \times \left(A^{lm}\; \zeta\; \mathbf{k} \right) + \frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \tag{B.4}$$

that is, in expanded form :

$$\begin{aligned} D^{\mathbf{U}}_u &= \frac{1}{e_1} \frac{\partial \left(A^{lm} \chi \right)}{\partial i} - \frac{1}{e_2} \frac{\partial \left(A^{lm} \zeta \right)}{\partial j} + \frac{1}{e_3} \frac{\partial u}{\partial k} \\ D^{\mathbf{U}}_v &= \frac{1}{e_2} \frac{\partial \left(A^{lm} \chi \right)}{\partial j} + \frac{1}{e_1} \frac{\partial \left(A^{lm} \zeta \right)}{\partial i} + \frac{1}{e_3} \frac{\partial v}{\partial k} \end{aligned}$$

Note Bene : introducing a rotation in (B.4) does not lead to a useful expression for the iso/diapycnal Laplacian operator in the z-coordinate. Similarly, we did not found an expression of practical use for the geopotential horizontal/vertical Laplacian operator in the s-coordinate. Generally, (B.4) is used in both z- and s-coordinate systems, that is a Laplacian diffusion is applied on momentum along the coordinate directions.

Discrete Invariants of the Equations

Contents

I'm writting this appendix. It will be available in a forthcoming release of the documentation

C.1 Conservation Properties on Ocean Dynamics

First, the boundary condition on the vertical velocity (no flux through the surface and the bottom) is established for the discrete set of momentum equations. Then, it is shown that the non-linear terms of the momentum equation are written such that the potential enstrophy of a horizontally non-divergent flow is preserved while all the other non-diffusive terms preserve the kinetic energy ; in practice the energy is also preserved. In addition, an option is also offered for the vorticity term discretization which provides a total kinetic energy conserving discretization for that term.

Nota Bene : these properties are established here in the rigid-lid case and for the 2nd order centered scheme. A forthcoming update will be their generalisation to the free surface case and higher order scheme.

C.1.1 Bottom Boundary Condition on Vertical Velocity Field

The discrete set of momentum equations used in the rigid-lid approximation automatically satisfies the surface and bottom boundary conditions (no flux through the surface and the bottom : $w_{surface} = w_{bottom} = 0$). Indeed, taking the discrete horizontal divergence of the vertical sum of the horizontal momentum equations (! ! !Eqs. (II.2.1) and (II.2.2) ! ! !) weighted by the vertical scale factors, it becomes :

$$\frac{\partial}{\partial t}\left(\sum_k \chi\right) \equiv \frac{\partial}{\partial t}\left(w_{surface} - w_{bottom}\right)$$

$$\equiv \frac{1}{e_{1T}\, e_{2T}\, e_{3T}} \left\{ \quad \delta_i \left[e_{2u}\, H_u \left(M_u - M_u - \frac{1}{H_u\, e_{2u}} \delta_j \left[\partial_t\, \psi \right] \right) \right] \right. \\ \left. + \delta_j \left[e_{1v}\, H_v \left(M_v - M_v - \frac{1}{H_v\, e_{1v}} \delta_i \left[\partial_i\, \psi \right] \right) \right] \right\}$$

$$\equiv \frac{1}{e_{1T}\, e_{2T}\, e_{3T}} \left\{ -\delta_i \Big[\delta_j \left[\partial_t \psi \right] \Big] + \delta_j \Big[\delta_i \left[\partial_t \psi \right] \Big] \right\} \equiv 0$$

The surface boundary condition associated with the rigid lid approximation ($w_{surface} = 0$) is imposed in the computation of the vertical velocity (! ! ! II.2.5 ! ! ! !). Therefore, it turns out to be :

$$\frac{\partial}{\partial t} w_{bottom} \equiv 0$$

As the bottom velocity is initially set to zero, it remains zero all the time. Symmetrically, if $w_{bottom} = 0$ is used in the computation of the vertical velocity (upward integral of the horizontal divergence), the same computation leads to $w_{surface} = 0$ as soon as the surface vertical velocity is initially set to zero.

C.1.2 Coriolis and advection terms : vector invariant form

Vorticity Term

Potential vorticity is located at f-points and defined as : ζ/e_{3f}. The standard discrete formulation of the relative vorticity term obviously conserves potential vorticity (ENS scheme). It also conserves the potential enstrophy for a horizontally non-divergent flow (i.e. χ=0) but not the total kinetic energy. Indeed, using the symmetry or skew symmetry properties of the operators (Eqs (3.10) and (3.9)), it can be shown that :

$$\int_D \zeta/e_3 \;\; \mathbf{k} \cdot \frac{1}{e_3} \nabla \times \left(\zeta \; \mathbf{k} \times \mathbf{U}_h \right) \; dv \equiv 0 \qquad \text{(C.1)}$$

where $dv = e_1\, e_2\, e_3 \; di\, dj\, dk$ is the volume element. Indeed, using (5.1), the discrete form of the right hand side of (C.1) can be transformed as follow :

$$\begin{aligned}
&\int_D \zeta/e_3 \;\; \mathbf{k} \cdot \frac{1}{e_3} \nabla \times \left(\zeta \; \mathbf{k} \times \mathbf{U}_h \right) \; dv \\
&\equiv \sum_{i,j,k} \frac{\zeta/e_{3f}}{e_{1f}\, e_{2f}\, e_{3f}} \biggl\{ \quad \delta_{i+1/2} \left[-\overline{\left(\zeta/e_{3f}\right)}^{\,i} \; \overline{\overline{\left(e_{2u}\, e_{3u}\, u\right)}}^{\,i,j+1/2} \right] \\
&\qquad\qquad\qquad - \delta_{j+1/2} \left[\quad \overline{\left(\zeta/e_{3f}\right)}^{\,j} \; \overline{\overline{\left(e_{1v}\, e_{3v}\, v\right)}}^{\,i+1/2,j} \right] \quad \biggr\} \; e_{1f}\, e_{2f}\, e_{3f} \\
&\equiv \sum_{i,j,k} \left\{ \delta_i \left[\frac{\zeta}{e_{3f}} \right] \; \overline{\left(\frac{\zeta}{e_{3f}} \right)}^{\,i} \; \overline{\overline{\left(e_{1u}\, e_{3u}\, u\right)}}^{\,i,j+1/2} + \delta_j \left[\frac{\zeta}{e_{3f}} \right] \; \overline{\left(\frac{\zeta}{e_{3f}} \right)}^{\,j} \; \overline{\overline{\left(e_{2v}\, e_{3v}\, v\right)}}^{\,i+1/2,j} \right\} \\
&\equiv \frac{1}{2} \sum_{i,j,k} \left\{ \delta_i \left[\left(\frac{\zeta}{e_{3f}} \right)^2 \right] \overline{\overline{\left(e_{2u}\, e_{3u}\, u\right)}}^{\,i,j+1/2} + \delta_j \left[\left(\zeta/e_{3f}\right)^2 \right] \overline{\overline{\left(e_{1v}\, e_{3v}\, v\right)}}^{\,i+1/2,j} \right\} \\
&\equiv -\frac{1}{2} \sum_{i,j,k} \left(\frac{\zeta}{e_{3f}} \right)^2 \; \left\{ \delta_{i+1/2} \left[\overline{\overline{\left(e_{2u}\, e_{3u}\, u\right)}}^{\,i,j+1/2} \right] + \delta_{j+1/2} \left[\overline{\overline{\left(e_{1v}\, e_{3v}\, v\right)}}^{\,i+1/2,j} \right] \right\}
\end{aligned}$$

Since $\overline{\cdot}$ and δ operators commute : $\delta_{i+1/2} \left[\overline{a}^{\,i} \right] = \overline{\delta_i \left[a \right]}^{\,i+1/2}$, and introducing the horizontal divergence χ, it becomes :

$$\equiv \sum_{i,j,k} -\frac{1}{2} \left(\frac{\zeta}{e_{3f}} \right)^2 \; \overline{\overline{e_{1T}\, e_{2T}\, e_{3T}\, \chi}}^{\,i+1/2,j+1/2} \quad \equiv 0$$

Note that the derivation is demonstrated here for the relative potential vorticity but it applies also to the planetary (f/e_3) and the total potential vorticity ($(\zeta + f)/e_3$). Another formulation of the two components of the vorticity term is optionally offered (ENE scheme) :

$$-\zeta\;\mathbf{k}\times\mathbf{U}_h \equiv \left(\begin{array}{c} +\frac{1}{e_{1u}}\;\overline{(\zeta/e_{3f})\;\overline{(e_{1v}\,e_{3v}\,v)}^{\,i+1/2}}^{\,j} \\ -\frac{1}{e_{2v}}\;\overline{(\zeta/e_{3f})\;\overline{(e_{2u}\,e_{3u}\,u)}^{\,j+1/2}}^{\,i} \end{array}\right)$$

This formulation does not conserve the enstrophy but it does conserve the total kinetic energy. It is also possible to mix the two formulations in order to conserve enstrophy on the relative vorticity term and energy on the Coriolis term.

$$\begin{aligned}
&\int_D -\mathbf{U}_h\cdot\left(\zeta\;\mathbf{k}\times\mathbf{U}_h\right)\;dv\\
&\equiv\sum_{i,j,k}\left\{ \overline{\left(\frac{\zeta}{e_{3f}}\right)\overline{(e_{1v}e_{3v}v)}^{\,i+1/2}}^{\,j}\; e_{2u}e_{3u}u - \overline{\left(\frac{\zeta}{e_{3f}}\right)\overline{(e_{2u}e_{3u}u)}^{\,j+1/2}}^{\,i}\; e_{1v}e_{3v}v\right\}\\
&\equiv\sum_{i,j,k}\frac{\zeta}{e_{3f}}\left\{\overline{(e_{1v}e_{3v}v)}^{\,i+1/2}\;\overline{(e_{2u}e_{3u}u)}^{\,j+1/2} - \overline{(e_{2u}e_{3u}u)}^{\,j+1/2}\;\overline{(e_{1v}e_{3v}v)}^{\,i+1/2}\right\}\equiv 0
\end{aligned}$$

Gradient of Kinetic Energy / Vertical Advection

The change of Kinetic Energy (KE) due to the vertical advection is exactly balanced by the change of KE due to the horizontal gradient of KE :

$$\int_D \mathbf{U}_h\cdot\nabla_h\left(\frac{1}{2}\;{\mathbf{U}_h}^2\right)\;dv = -\int_D \mathbf{U}_h\cdot w\frac{\partial\mathbf{U}_h}{\partial k}\;dv$$

Indeed, using successively (3.9) ($i.e.$ the skew symmetry property of the δ operator) and the incompressibility, then (3.9) again, then the commutativity of operators $\overline{\cdot}$ and δ, and finally (3.10) ($i.e.$ the symmetry property of the $\overline{\cdot}$ operator) applied in the horizontal and vertical directions, it becomes :

$$\begin{aligned}
&\int_D \mathbf{U}_h\cdot\nabla_h\left(\frac{1}{2}\;{\mathbf{U}_h}^2\right)\;dv\\
&\equiv\frac{1}{2}\sum_{i,j,k}\left\{\frac{1}{e_{1u}}\delta_{i+1/2}\left[\overline{u^2}^{\,i}+\overline{v^2}^{\,j}\right]u\,e_{1u}e_{2u}e_{3u}+\frac{1}{e_{2v}}\delta_{j+1/2}\left[\overline{u^2}^{\,i}+\overline{v^2}^{\,j}\right]v\,e_{1v}e_{2v}e_{3v}\right\}\\
&\equiv\frac{1}{2}\sum_{i,j,k}\left(\overline{u^2}^{\,i}+\overline{v^2}^{\,j}\right)\;\delta_k\left[e_{1T}\,e_{2T}\,w\right]\quad\equiv-\frac{1}{2}\sum_{i,j,k}\delta_{k+1/2}\left[\overline{u^2}^{\,i}+\overline{v^2}^{\,j}\right]\;e_{1v}\,e_{2v}\,w\\
&\equiv\frac{1}{2}\sum_{i,j,k}\left(\overline{\delta_{k+1/2}\left[u^2\right]}^{\,i}+\overline{\delta_{k+1/2}\left[v^2\right]}^{\,j}\right)\;e_{1T}\,e_{2T}\,w\\
&\equiv\frac{1}{2}\sum_{i,j,k}\left\{\overline{e_{1T}\,e_{2T}\,w}^{\,i+1/2}\;2\overline{u}^{\,k+1/2}\;\delta_{k+1/2}\left[u\right]+\overline{e_{1T}\,e_{2T}\,w}^{\,j+1/2}\;2\overline{v}^{\,k+1/2}\;\delta_{k+1/2}\left[v\right]\right\}
\end{aligned}$$

$$\begin{aligned}
&\equiv - \sum_{i,j,k} \Biggl\{ \;\; \frac{1}{b_u} \; \overline{ \left\{ \overline{e_{1T}\, e_{2T}\, w}^{\,i+1/2} \, \delta_{k+1/2}\left[u \right] \right\} }^{\,k} \; u \; e_{1u} \, e_{2u} \, e_{3u} \\
&\qquad\qquad + \frac{1}{b_v} \; \overline{ \left\{ \overline{e_{1T}\, e_{2T}\, w}^{\,j+1/2} \delta_{k+1/2}\left[v \right] \right\} }^{\,k} \; v \; e_{1v} \, e_{2v} \, e_{3v} \;\; \Biggr\} \\
&\equiv - \int\limits_D \textbf{U}_h \cdot w \frac{\partial \textbf{U}_h}{\partial k} \; dv
\end{aligned}$$

The main point here is that the satisfaction of this property links the choice of the discrete formulation of the vertical advection and of the horizontal gradient of KE. Choosing one imposes the other. For example KE can also be discretized as $1/2\,(\overline{u}^{\,i^2} + \overline{v}^{\,j^2})$. This leads to the following expression for the vertical advection :

$$\frac{1}{e_3}\; w\; \frac{\partial \textbf{U}_h}{\partial k} \equiv \left(\begin{matrix} \frac{1}{e_{1u}\,e_{2u}\,e_{3u}} \; \overline{\overline{ e_{1T}\,e_{2T}\; w \; \delta_{k+1/2}\left[\overline{u}^{\,i+1/2} \right] }}^{\,i+1/2,k} \\ \frac{1}{e_{1v}\,e_{2v}\,e_{3v}} \; \overline{\overline{ e_{1T}\,e_{2T}\; w \; \delta_{k+1/2}\left[\overline{v}^{\,j+1/2} \right] }}^{\,j+1/2,k} \end{matrix} \right)$$

a formulation that requires an additional horizontal mean in contrast with the one used in NEMO. Nine velocity points have to be used instead of 3. This is the reason why it has not been chosen.

C.1.3 Coriolis and advection terms : flux form

Coriolis plus “metric” Term

In flux from the vorticity term reduces to a Coriolis term in which the Coriolis parameter has been modified to account for the “metric” term. This altered Coriolis parameter is discretised at an f-point. It is given by :

$$f+\frac{1}{e_1 e_2} \left(v\frac{\partial e_2}{\partial i} - u\frac{\partial e_1}{\partial j} \right) \;\equiv \; f+\frac{1}{e_{1f}\,e_{2f}} \left(\overline{v}^{\,i+1/2}\delta_{i+1/2}\left[e_{2u} \right] - \overline{u}^{\,j+1/2}\delta_{j+1/2}\left[e_{1u} \right] \right)$$

The ENE scheme is then applied to obtain the vorticity term in flux form. It therefore conserves the total KE. The derivation is the same as for the vorticity term in the vector invariant form (§C.1.2).

Flux form advection

The flux form operator of the momentum advection is evaluated using a centered second order finite difference scheme. Because of the flux form, the discrete operator does not contribute to the global budget of linear momentum. Because of the centered second order scheme, it conserves the horizontal kinetic energy, that is :

$$\int_D \textbf{U}_h \cdot \left(\begin{matrix} \nabla \cdot \left(\textbf{U}\,u \right) \\ \nabla \cdot \left(\textbf{U}\,v \right) \end{matrix} \right) \; dv = \;0 \tag{C.2}$$

Let us demonstrate this property for the first term of the scalar product ($i.e.$ considering just the the terms associated with the i-component of the advection) :

$$\begin{aligned}
&\int_D u \cdot \nabla \cdot (\mathbf{U}\, u) \;\; dv \\
&\equiv \sum_{i,j,k} \left\{ \frac{1}{e_{1u}\, e_{2u}\, e_{3u}} \left(\delta_{i+1/2} \left[\overline{e_{2u}\, e_{3u}\, u}^{\,i} \; \overline{u}^{\,i} \right] + \delta_j \left[\overline{e_{1u}\, e_{3u}\, v}^{\,i+1/2} \; \overline{u}^{\,j+1/2} \right] \right.\right. \\
&\qquad\qquad \left.\left. + \delta_k \left[\overline{e_{1w}\, e_{2w}\, w}^{\,i+1/2} \; \overline{u}^{\,k+1/2} \right] \right) \right\} e_{1u}\, e_{2u}\, e_{3u}\; u \\
&\equiv \sum_{i,j,k} \left\{ \delta_{i+1/2} \left[\overline{e_{2u}\, e_{3u}\, u}^{\,i} \; \overline{u}^{\,i} \right] + \delta_j \left[\overline{e_{1u}\, e_{3u}\, v}^{\,i+1/2} \; \overline{u}^{\,j+1/2} \right] \right. \\
&\qquad\qquad \left. + \delta_k \left[\overline{e_{1w}\, e_{2w}\, w}^{\,i+12} \; \overline{u}^{\,k+1/2} \right] \right\} \; u \\
&\equiv - \sum_{i,j,k} \left\{ \overline{e_{2u}\, e_{3u}\, u}^{\,i} \; \overline{u}^{\,i} \delta_i \left[u \right] + \overline{e_{1u}\, e_{3u}\, v}^{\,i+1/2} \; \overline{u}^{\,j+1/2} \delta_{j+1/2} \left[u \right] \right. \\
&\qquad\qquad \left. + \overline{e_{1w}\, e_{2w}\, w}^{\,i+1/2} \; \overline{u}^{\,k+1/2} \delta_{k+1/2} \left[u \right] \right\} \\
&\equiv - \sum_{i,j,k} \left\{ \overline{e_{2u}\, e_{3u}\, u}^{\,i} \delta_i \left[u^2 \right] + \overline{e_{1u}\, e_{3u}\, v}^{\,i+1/2} \delta_{j+/2} \left[u^2 \right] + \overline{e_{1w}\, e_{2w}\, w}^{\,i+1/2} \delta_{k+1/2} \left[u^2 \right] \right\} \\
&\equiv \sum_{i,j,k} \left\{ e_{2u}\, e_{3u}\, u \;\delta_{i+1/2} \left[\overline{u^2}^{\,i} \right] + e_{1u}\, e_{3u}\, v \;\delta_{j+1/2} \left[\overline{u^2}^{\,i} \right] + e_{1w}\, e_{2w}\, w \;\delta_{k+1/2} \left[\overline{u^2}^{\,i} \right] \right\} \\
&\equiv \sum_{i,j,k} \overline{u^2}^{\,i} \left\{ \delta_{i+1/2} \left[e_{2u}\, e_{3u}\, u \right] + \delta_{j+1/2} \left[e_{1u}\, e_{3u}\, v \right] + \delta_{k+1/2} \left[e_{1w}\, e_{2w}\, w \right] \right\} \quad \equiv 0
\end{aligned}$$

When the UBS scheme is used to evaluate the flux form momentum advection, the discrete operator does not contribute to the global budget of linear momentum (flux form). The horizontal kinetic energy is not conserved, but forced to decay ($i.e.$ the scheme is diffusive).

C.1.4 Hydrostatic Pressure Gradient Term

A pressure gradient has no contribution to the evolution of the vorticity as the curl of a gradient is zero. In the z-coordinate, this property is satisfied locally on a C-grid with 2nd order finite differences (property (3.7)). When the equation of state is linear ($i.e.$ when an advection-diffusion equation for density can be derived from those of temperature and salinity) the change of KE due to the work of pressure forces is balanced by the change of potential energy due to buoyancy forces :

$$\int_D -\frac{1}{\rho_o} \; \nabla p^h \Big|_z \cdot \mathbf{U}_h \;\; dv = \int_D \nabla \cdot \left(\rho\, \mathbf{U} \right) \;\; g \;\, z \;\; dv$$

This property can be satisfied in a discrete sense for both z- and s-coordinates. Indeed, defining the depth of a T-point, z_T, as the sum of the vertical scale factors at w-points starting from the surface, the work of pressure forces can be written as :

$$\int_D -\frac{1}{\rho_o} \left. \nabla p^h \right|_z \cdot \mathbf{U}_h \; dv$$

$$\equiv \sum_{i,j,k} \Biggl\{ -\frac{1}{\rho_o e_{1u}} \Bigl(\delta_{i+1/2}\left[p^h \right] - g\;\overline{\rho}^{\,i+1/2}\;\delta_{i+1/2}\left[z_T \right] \Bigr)\; u\; e_{1u}\, e_{2u}\, e_{3u}$$

$$- \frac{1}{\rho_o e_{2v}} \Bigl(\delta_{j+1/2}\left[p^h \right] - g\;\overline{\rho}^{\,j+1/2} \delta_{j+1/2}\left[z_T \right] \Bigr)\; v\; e_{1v}\, e_{2v}\, e_{3v} \Biggr\}$$

Using (3.9), $i.e.$ the skew symmetry property of the δ operator, (5.31), the continuity equation), and (5.16), the hydrostatic equation in the s-coordinate, it becomes :

$$\equiv \frac{1}{\rho_o} \sum_{i,j,k} \Biggl\{ e_{2u}\, e_{3u}\; u\; g\; \overline{\rho}^{\,i+1/2}\; \delta_{i+1/2}[z_T] + e_{1v}\, e_{3v}\; v\; g\; \overline{\rho}^{\,j+1/2}\; \delta_{j+1/2}[z_T]$$

$$+ \Bigl(\delta_i[e_{2u}\, e_{3u}\, u] + \delta_j[e_{1v}\, e_{3v}\, v] \Bigr)\; p^h \Biggr\}$$

$$\equiv \frac{1}{\rho_o} \sum_{i,j,k} \Biggl\{ e_{2u}\, e_{3u}\; u\; g\; \overline{\rho}^{\,i+1/2} \delta_{i+1/2}\left[z_T \right] + e_{1v}\, e_{3v}\; v\; g\; \overline{\rho}^{\,j+1/2} \delta_{j+1/2}\left[z_T \right]$$

$$- \delta_k \left[e_{1w} e_{2w}\; w \right]\; p^h \Biggr\}$$

$$\equiv \frac{1}{\rho_o} \sum_{i,j,k} \Biggl\{ e_{2u}\, e_{3u}\; u\; g\; \overline{\rho}^{\,i+1/2}\; \delta_{i+1/2}\left[z_T \right] + e_{1v}\, e_{3v}\; v\; g\; \overline{\rho}^{\,j+1/2}\; \delta_{j+1/2}\left[z_T \right]$$

$$+ e_{1w} e_{2w}\; w\; \delta_{k+1/2}\left[p_h \right] \Biggr\}$$

$$\equiv \frac{g}{\rho_o} \sum_{i,j,k} \Biggl\{ e_{2u}\, e_{3u}\; u\; \overline{\rho}^{\,i+1/2}\; \delta_{i+1/2}\left[z_T \right] + e_{1v}\, e_{3v}\; v\; \overline{\rho}^{\,j+1/2}\; \delta_{j+1/2}\left[z_T \right]$$

$$- e_{1w} e_{2w}\; w\; e_{3w} \overline{\rho}^{\,k+1/2} \Biggr\}$$

noting that by definition of z_T, $\delta_{k+1/2}\left[z_T \right] \equiv -e_{3w}$, thus :

$$\equiv \frac{g}{\rho_o} \sum_{i,j,k} \Biggl\{ e_{2u}\, e_{3u}\; u\; \overline{\rho}^{\,i+1/2}\; \delta_{i+1/2}\left[z_T \right] + e_{1v}\, e_{3v}\; v\; \overline{\rho}^{\,j+1/2} \delta_{j+1/2}\left[z_T \right]$$

$$+ e_{1w} e_{2w}\; w\; \overline{\rho}^{\,k+1/2}\; \delta_{k+1/2}\left[z_T \right] \Biggr\}$$

Using (3.9), it becomes :

$$\equiv -\frac{g}{\rho_o}\sum_{i,j,k} z_T \left\{ \delta_i\left[e_{2u}\,e_{3u}\,u\,\overline{\rho}^{\,i+1/2}\right] + \delta_j\left[e_{1v}\,e_{3v}\,v\,\overline{\rho}^{\,j+1/2}\right] + \delta_k\left[e_{1w}e_{2w}\,w\,\overline{\rho}^{\,k+1/2}\right]\right\}$$

$$\equiv -\int_D \nabla\cdot\left(\rho\,\mathbf{U}\right)\; g\; z\;\; dv$$

Note that this property strongly constrains the discrete expression of both the depth of $T-$points and of the term added to the pressure gradient in the s-coordinate. Nevertheless, it is almost never satisfied since a linear equation of state is rarely used.

C.1.5 Surface Pressure Gradient Term

The surface pressure gradient has no contribution to the evolution of the vorticity. This property is trivially satisfied locally since the equation verified by ψ has been derived from the discrete formulation of the momentum equation and of the curl. But it has to be noted that since the elliptic equation satisfied by ψ is solved numerically by an iterative solver (preconditioned conjugate gradient or successive over relaxation), the property is only satisfied at the precision requested for the solver used.

With the rigid-lid approximation, the change of KE due to the work of surface pressure forces is exactly zero. This is satisfied in discrete form, at the precision requested for the elliptic solver used to solve this equation. This can be demonstrated as follows :

$$\int_D -\frac{1}{\rho_o}\nabla_h\left(p_s\right)\cdot\mathbf{U}_h\;dv \equiv \sum_{i,j,k}\left\{\ \left(-M_u - \frac{1}{H_u\,e_{2u}}\delta_j\left[\partial_t\psi\right]\right)\ u\;e_{1u}\,e_{2u}\,e_{3u} + \left(-M_v + \frac{1}{H_v\,e_{1v}}\delta_i\left[\partial_t\psi\right]\right)\ v\;e_{1v}\,e_{2v}\,e_{3v}\ \right\}$$

$$\equiv \sum_{i,j}\left\{\ \left(-M_u - \frac{1}{H_u\,e_{2u}}\delta_j\left[\partial_t\psi\right]\right)\left(\sum_k u\;e_{3u}\right)\ e_{1u}\,e_{2u} + \left(-M_v + \frac{1}{H_v\,e_{1v}}\delta_i\left[\partial_t\psi\right]\right)\left(\sum_k v\;e_{3v}\right)\ e_{1v}\,e_{2v}\ \right\}$$

using the relation between ψ and the vertical sum of the velocity, it becomes :

$$\equiv \sum_{i,j}\left\{\ \left(\ \ M_u + \frac{1}{H_u\,e_{2u}}\delta_j\left[\partial_t\psi\right]\right)\ e_{1u}\,\delta_j\left[\partial_t\psi\right] + \left(-M_v + \frac{1}{H_v\,e_{1v}}\delta_i\left[\partial_t\psi\right]\right)\ e_{2v}\,\delta_i\left[\partial_t\psi\right]\ \right\}$$

applying the adjoint of the δ operator, it is now :

$$\begin{aligned}
&\equiv \sum_{i,j} -\partial_t \psi \; \Biggl\{ \; \delta_{j+1/2}\left[e_{1u} M_u\right] - \delta_{i+1/2}\left[e_{1v} M_v\right] \\
&\qquad + \delta_{i+1/2}\left[\frac{e_{2v}}{H_v\, e_{2v}} \delta_i\left[\partial_t \psi\right]\right] + \delta_{j+1/2}\left[\frac{e_{1u}}{H_u\, e_{2u}} \delta_j\left[\partial_t \psi\right]\right]\Biggr\} \\
&\equiv 0
\end{aligned}$$

The last equality is obtained using (5.22), the discrete barotropic streamfunction time evolution equation. By the way, this shows that (5.22) is the only way to compute the streamfunction, otherwise the surface pressure forces will do work. Nevertheless, since the elliptic equation satisfied by ψ is solved numerically by an iterative solver, the property is only satisfied at the precision requested for the solver.

C.2 Conservation Properties on Tracers

All the numerical schemes used in NEMO are written such that the tracer content is conserved by the internal dynamics and physics (equations in flux form). For advection, only the CEN2 scheme ($i.e.$ 2^{nd} order finite different scheme) conserves the global variance of tracer. Nevertheless the other schemes ensure that the global variance decreases ($i.e.$ they are at least slightly diffusive). For diffusion, all the schemes ensure the decrease of the total tracer variance, except the iso-neutral operator. There is generally no strict conservation of mass, as the equation of state is non linear with respect to T and S. In practice, the mass is conserved to a very high accuracy.

C.2.1 Advection Term

Whatever the advection scheme considered it conserves of the tracer content as all the scheme are written in flux form. Let τ be the tracer interpolated at velocity point (whatever the interpolation is). The conservation of the tracer content is obtained as follows :

$$\begin{aligned}
&\int_D \nabla \cdot (T \mathbf{U}) \; dv \\
&\equiv \sum_{i,j,k} \biggl\{ \frac{1}{e_{1T}\, e_{2T}\, e_{3T}} \left(\delta_i\left[e_{2u}\, e_{3u}\; u\; \tau_u\right] + \delta_j\left[e_{1v}\, e_{3v}\; v\; \tau_v\right]\right) \\
&\qquad\qquad + \frac{1}{e_{3T}} \delta_k\left[w\; \tau\right] \biggr\} e_{1T}\, e_{2T}\, e_{3T} \\
&\equiv \sum_{i,j,k} \left\{ \delta_i\left[e_{2u}\, e_{3u}\, \overline{T}^{\,i+1/2}\, u\right] + \delta_j\left[e_{1v}\, e_{3v}\, \overline{T}^{\,j+1/2}\, v\right] + \delta_k\left[e_{1T}\, e_{2T}\, \overline{T}^{\,k+1/2}\, w\right] \right\} \\
&\equiv 0
\end{aligned}$$

The conservation of the variance of tracer can be achieved only with the CEN2 scheme. It can be demonstarted as follows :

$$\begin{aligned}
&\int_D T\ \nabla\cdot(T\ \mathbf{U})\ dv \\
&\equiv \sum_{i,j,k} T\ \left\{ \delta_i\left[e_{2u}\,e_{3u}\overline{T}^{\,i+1/2}\ u\right] + \delta_j\left[e_{1v}\,e_{3v}\overline{T}^{\,j+1/2}\ v\right] + \delta_k\left[e_{1T}\,e_{2T}\overline{T}^{\,k+1/2} w\right]\right\} \\
&\equiv \sum_{i,j,k} \Big\{ -e_{2u}\,e_{3u}\overline{T}^{\,i+1/2}\,u\,\delta_{i+1/2}\left[T\right] - e_{1v}\,e_{3v}\overline{T}^{\,j+1/2}\,v\ \delta_{j+1/2}\left[T\right] \\
&\qquad\qquad - e_{1T}\,e_{2T}\overline{T}^{\,k+1/2} w\ \delta_{k+1/2}\left[T\right]\Big\} \\
&\equiv -\frac{1}{2}\sum_{i,j,k}\left\{ e_{2u}\,e_{3u}\ u\ \delta_{i+1/2}\left[T^2\right] + e_{1v}\,e_{3v}\ v\ \delta_{j+1/2}\left[T^2\right] + e_{1T}\,e_{2T}\ w\ \delta_{k+1/2}\left[T^2\right]\right\} \\
&\equiv \frac{1}{2}\sum_{i,j,k} T^2\left\{\delta_i\left[e_{2u}\,e_{3u}\,u\right] + \delta_j\left[e_{1v}\,e_{3v}\,v\right] + \delta_k\left[e_{1T}\,e_{2T}\,w\right]\right\} \qquad \equiv 0
\end{aligned}$$

C.3 Conservation Properties on Lateral Momentum Physics

The discrete formulation of the horizontal diffusion of momentum ensures the conservation of potential vorticity and the horizontal divergence, and the dissipation of the square of these quantities (i.e. enstrophy and the variance of the horizontal divergence) as well as the dissipation of the horizontal kinetic energy. In particular, when the eddy coefficients are horizontally uniform, it ensures a complete separation of vorticity and horizontal divergence fields, so that diffusion (dissipation) of vorticity (enstrophy) does not generate horizontal divergence (variance of the horizontal divergence) and *vice versa*.

These properties of the horizontal diffusion operator are a direct consequence of properties (3.7) and (3.8). When the vertical curl of the horizontal diffusion of momentum (discrete sense) is taken, the term associated with the horizontal gradient of the divergence is locally zero.

C.3.1 Conservation of Potential Vorticity

The lateral momentum diffusion term conserves the potential vorticity :

$$\int_D \frac{1}{e_3}\mathbf{k}\cdot\nabla\times\left[\nabla_h\left(A^{\,lm}\,\chi\right) - \nabla_h\times\left(A^{\,lm}\,\zeta\ \mathbf{k}\right)\right]\ dv = 0$$

$$= \int_D -\frac{1}{e_3}\mathbf{k}\cdot\nabla\times\left[\nabla_h\times\left(A^{\,lm}\,\zeta\ \mathbf{k}\right)\right]\ dv$$

$$\equiv \sum_{i,j} \left\{ \delta_{i+1/2} \left[\frac{e_{2v}}{e_{1v}\,e_{3v}} \delta_i \left[A_f^{lm} e_{3f} \zeta \right] \right] + \delta_{j+1/2} \left[\frac{e_{1u}}{e_{2u}\,e_{3u}} \delta_j \left[A_f^{lm} e_{3f} \zeta \right] \right] \right\}$$

Using (3.9), it follows :

$$\equiv \sum_{i,j,k} - \left\{ \frac{e_{2v}}{e_{1v}\,e_{3v}} \delta_i \left[A_f^{lm} e_{3f} \zeta \right] \; \delta_i \left[1 \right] + \frac{e_{1u}}{e_{2u}\,e_{3u}} \delta_j \left[A_f^{lm} e_{3f} \zeta \right] \; \delta_j \left[1 \right] \right\} \qquad \equiv 0$$

C.3.2 Dissipation of Horizontal Kinetic Energy

The lateral momentum diffusion term dissipates the horizontal kinetic energy :

$$\int_D \mathbf{U}_h \cdot \left[\nabla_h \left(A^{lm} \; \chi \right) - \nabla_h \times \left(A^{lm} \; \zeta \; \mathbf{k} \right) \right] \; dv$$

$$\begin{aligned} \equiv \sum_{i,j,k} & \left\{ \frac{1}{e_{1u}} \delta_{i+1/2} \left[A_T^{lm} \chi \right] - \frac{1}{e_{2u}\,e_{3u}} \delta_j \left[A_f^{lm} e_{3f} \zeta \right] \right\} \; e_{1u}\,e_{2u}\,e_{3u} \; u \\ & + \left\{ \frac{1}{e_{2u}} \delta_{j+1/2} \left[A_T^{lm} \chi \right] + \frac{1}{e_{1v}\,e_{3v}} \delta_i \left[A_f^{lm} e_{3f} \zeta \right] \right\} \; e_{1v}\,e_{2u}\,e_{3v} \; v \end{aligned}$$

$$\begin{aligned} \equiv \sum_{i,j,k} & \left\{ e_{2u}\,e_{3u} \; u \; \delta_{i+1/2} \left[A_T^{lm} \chi \right] - e_{1u} \; u \; \delta_j \left[A_f^{lm} e_{3f} \zeta \right] \right\} \\ & + \left\{ e_{1v}\,e_{3v} \; v \; \delta_{j+1/2} \left[A_T^{lm} \chi \right] + e_{2v} \; v \; \delta_i \left[A_f^{lm} e_{3f} \zeta \right] \right\} \end{aligned}$$

$$\begin{aligned} \equiv \sum_{i,j,k} & - \left(\delta_i \left[e_{2u}\,e_{3u} \; u \right] + \delta_j \left[e_{1v}\,e_{3v} \; v \right] \right) \; A_T^{lm} \chi \\ & - \left(\delta_{i+1/2} \left[e_{2v} \; v \right] - \delta_{j+1/2} \left[e_{1u} \; u \right] \right) \; A_f^{lm} e_{3f} \zeta \end{aligned}$$

$$\equiv \sum_{i,j,k} - A_T^{lm} \, \chi^2 \; e_{1T}\,e_{2T}\,e_{3T} - A_f^{lm} \, \zeta^2 \; e_{1f}\,e_{2f}\,e_{3f} \qquad \leq 0$$

C.3.3 Dissipation of Enstrophy

The lateral momentum diffusion term dissipates the enstrophy when the eddy coefficients are horizontally uniform :

$$\begin{aligned}
&\int_D \zeta \, \mathbf{k} \cdot \nabla \times \left[\nabla_h \left(A^{lm} \, \chi \right) - \nabla_h \times \left(A^{lm} \, \zeta \, \mathbf{k} \right) \right] \; dv \\
&= A^{lm} \int_D \zeta \mathbf{k} \cdot \nabla \times \left[\nabla_h \times (\zeta \, \mathbf{k}) \right] \; dv \\
&\equiv A^{lm} \sum_{i,j,k} \zeta \; e_{3f} \left\{ \delta_{i+1/2} \left[\frac{e_{2v}}{e_{1v} \, e_{3v}} \delta_i \left[e_{3f} \zeta \right] \right] + \delta_{j+1/2} \left[\frac{e_{1u}}{e_{2u} \, e_{3u}} \delta_j \left[e_{3f} \zeta \right] \right] \right\}
\end{aligned}$$

Using (3.9), it follows :

$$\begin{aligned}
&\equiv -A^{lm} \sum_{i,j,k} \left\{ \left(\frac{1}{e_{1v} \, e_{3v}} \delta_i \left[e_{3f} \zeta \right] \right)^2 e_{1v} \, e_{2v} \, e_{3v} + \left(\frac{1}{e_{2u} \, e_{3u}} \delta_j \left[e_{3f} \zeta \right] \right)^2 e_{1u} \, e_{2u} \, e_{3u} \right\} \\
&\leq \; 0
\end{aligned}$$

C.3.4 Conservation of Horizontal Divergence

When the horizontal divergence of the horizontal diffusion of momentum (discrete sense) is taken, the term associated with the vertical curl of the vorticity is zero locally, due to (! ! ! II.1.8 ! ! ! ! !). The resulting term conserves the χ and dissipates χ^2 when the eddy coefficients are horizontally uniform.

$$\begin{aligned}
&\int_D \nabla_h \cdot \left[\nabla_h \left(A^{lm} \, \chi \right) - \nabla_h \times \left(A^{lm} \, \zeta \, \mathbf{k} \right) \right] dv = \int_D \nabla_h \cdot \nabla_h \left(A^{lm} \, \chi \right) dv \\
&\equiv \sum_{i,j,k} \left\{ \delta_i \left[A_u^{lm} \frac{e_{2u} \, e_{3u}}{e_{1u}} \delta_{i+1/2} \left[\chi \right] \right] + \delta_j \left[A_v^{lm} \frac{e_{1v} \, e_{3v}}{e_{2v}} \delta_{j+1/2} \left[\chi \right] \right] \right\}
\end{aligned}$$

Using (3.9), it follows :

$$\equiv \sum_{i,j,k} - \left\{ \frac{e_{2u} \, e_{3u}}{e_{1u}} A_u^{lm} \delta_{i+1/2} \left[\chi \right] \delta_{i+1/2} \left[1 \right] + \frac{e_{1v} \, e_{3v}}{e_{2v}} A_v^{lm} \delta_{j+1/2} \left[\chi \right] \delta_{j+1/2} \left[1 \right] \right\} \; \equiv 0$$

C.3.5 Dissipation of Horizontal Divergence Variance

$$\int\limits_D \chi \, \nabla_h \cdot \left[\nabla_h \left(A^{lm} \, \chi \right) - \nabla_h \times \left(A^{lm} \, \zeta \, \mathbf{k} \right) \right] \, dv = A^{lm} \int\limits_D \chi \, \nabla_h \cdot \nabla_h \left(\chi \right) \, dv$$

$$\equiv A^{lm} \sum_{i,j,k} \frac{1}{e_{1T} \, e_{2T} \, e_{3T}} \chi \left\{ \delta_i \left[\frac{e_{2u} \, e_{3u}}{e_{1u}} \delta_{i+1/2} \left[\chi \right] \right] + \delta_j \left[\frac{e_{1v} \, e_{3v}}{e_{2v}} \delta_{j+1/2} \left[\chi \right] \right] \right\} \, e_{1T} \, e_{2T} \, e_{3T}$$

Using (3.9), it turns out to be :

$$\equiv -A^{lm} \sum_{i,j,k} \left\{ \left(\frac{1}{e_{1u}} \delta_{i+1/2} \left[\chi \right] \right)^2 e_{1u} \, e_{2u} \, e_{3u} + \left(\frac{1}{e_{2v}} \delta_{j+1/2} \left[\chi \right] \right)^2 e_{1v} \, e_{2v} \, e_{3v} \right\}$$
$$\leq 0$$

C.4 Conservation Properties on Vertical Momentum Physics

As for the lateral momentum physics, the continuous form of the vertical diffusion of momentum satisfies several integral constraints. The first two are associated with the conservation of momentum and the dissipation of horizontal kinetic energy :

$$\int\limits_D \frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \, dv \qquad = \vec{\mathbf{0}}$$

and

$$\int\limits_D \mathbf{U}_h \cdot \frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \, dv \quad \leq 0$$

The first property is obvious. The second results from :

$$\int\limits_D \mathbf{U}_h \cdot \frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \, dv$$

$$\equiv \sum_{i,j,k} \left(u \, \delta_k \left[\frac{A_u^{vm}}{e_{3uw}} \delta_{k+1/2} \left[u \right] \right] \, e_{1u} \, e_{2u} + v \, \delta_k \left[\frac{A_v^{vm}}{e_{3vw}} \delta_{k+1/2} \left[v \right] \right] \, e_{1v} \, e_{2v} \right)$$

since the horizontal scale factor does not depend on k, it follows :

$$\equiv -\sum_{i,j,k} \left(\frac{A_u^{vm}}{e_{3uw}} \left(\delta_{k+1/2}\left[u\right]\right)^2 \; e_{1u}\, e_{2u} + \frac{A_v^{vm}}{e_{3vw}} \left(\delta_{k+1/2}\left[v\right]\right)^2 \; e_{1v}\, e_{2v} \right) \quad \leq 0$$

The vorticity is also conserved. Indeed :

$$\int\limits_D \frac{1}{e_3} \mathbf{k} \cdot \nabla \times \left(\frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \right) \; dv$$

$$\begin{aligned} \equiv \sum_{i,j,k} \frac{1}{e_{3f}} \; \frac{1}{e_{1f}\, e_{2f}} \Bigg\{ \quad & \delta_{i+1/2} \left(\frac{e_{2v}}{e_{3v}} \delta_k \left[\frac{1}{e_{3vw}} \delta_{k+1/2}\left[v\right] \right] \right) \\ & - \delta_{j+1/2} \left(\frac{e_{1u}}{e_{3u}} \delta_k \left[\frac{1}{e_{3uw}} \delta_{k+1/2}\left[u\right] \right] \right) \Bigg\} \; e_{1f}\, e_{2f}\, e_{3f} \;\; \equiv 0 \end{aligned}$$

If the vertical diffusion coefficient is uniform over the whole domain, the enstrophy is dissipated, $i.e.$

$$\int\limits_D \zeta \, \mathbf{k} \cdot \nabla \times \left(\frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \right) \; dv = 0$$

This property is only satisfied in z-coordinates :

$$\int\limits_D \zeta \, \mathbf{k} \cdot \nabla \times \left(\frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \right) \; dv$$

$$\begin{aligned} \equiv \sum_{i,j,k} \zeta \; e_{3f} \; \Bigg\{ \quad & \delta_{i+1/2} \left(\frac{e_{2v}}{e_{3v}} \delta_k \left[\frac{A_v^{vm}}{e_{3vw}} \delta_{k+1/2}\left[v\right] \right] \right) \\ & - \delta_{j+1/2} \left(\frac{e_{1u}}{e_{3u}} \delta_k \left[\frac{A_u^{vm}}{e_{3uw}} \delta_{k+1/2}\left[u\right] \right] \right) \Bigg\} \end{aligned}$$

$$\equiv \sum_{i,j,k} \zeta \; e_{3f} \left\{ \quad \frac{1}{e_{3v}} \delta_k \left[\frac{A_v^{vm}}{e_{3vw}} \delta_{k+1/2} \left[\delta_{i+1/2} \left[e_{2v} \, v \right] \right] \right] \right.$$
$$\left. - \frac{1}{e_{3u}} \delta_k \left[\frac{A_u^{vm}}{e_{3uw}} \delta_{k+1/2} \left[\delta_{j+1/2} \left[e_{1u} \, u \right] \right] \right] \right\}$$

Using the fact that the vertical diffusion coefficients are uniform, and that in z-coordinate, the vertical scale factors do not depend on i and j so that : $e_{3f} = e_{3u} = e_{3v} = e_{3T}$ and $e_{3w} = e_{3uw} = e_{3vw}$, it follows :

$$\equiv A^{vm} \sum_{i,j,k} \zeta \; \delta_k \left[\frac{1}{e_{3w}} \delta_{k+1/2} \Big[\delta_{i+1/2} \left[e_{2v} \, v \right] - \delta_{j+1/2} \left[e_{1u} \, u \right] \Big] \right]$$

$$\equiv -A^{vm} \sum_{i,j,k} \frac{1}{e_{3w}} \left(\delta_{k+1/2} \left[\zeta \right] \right)^2 \; e_{1f} \, e_{2f} \;\; \leq 0$$

Similarly, the horizontal divergence is obviously conserved :

$$\int\limits_D \nabla \cdot \left(\frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \right) \; dv = 0$$

and the square of the horizontal divergence decreases ($i.e.$ the horizontal divergence is dissipated) if the vertical diffusion coefficient is uniform over the whole domain :

$$\int\limits_D \chi \; \nabla \cdot \left(\frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \right) \; dv = 0$$

This property is only satisfied in the z-coordinate :

$$\int\limits_D \chi \; \nabla \cdot \left(\frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \right) \; dv$$

$$\equiv \sum_{i,j,k} \frac{\chi}{e_{1T}\, e_{2T}} \left\{ \quad \delta_{i+1/2}\left(\frac{e_{2u}}{e_{3u}} \delta_k\left[\frac{A_u^{vm}}{e_{3uw}} \delta_{k+1/2}[u] \right] \right) \right.$$
$$\left. + \delta_{j+1/2}\left(\frac{e_{1v}}{e_{3v}} \delta_k\left[\frac{A_v^{vm}}{e_{3vw}} \delta_{k+1/2}[v] \right] \right) \right\} e_{1T}\, e_{2T}\, e_{3T}$$

$$\equiv A^{vm} \sum_{i,j,k} \chi \left\{ \quad \delta_{i+1/2}\left(\delta_k\left[\frac{1}{e_{3uw}} \delta_{k+1/2}[e_{2u}\, u] \right] \right) \right.$$
$$\left. + \delta_{j+1/2}\left(\delta_k\left[\frac{1}{e_{3vw}} \delta_{k+1/2}[e_{1v}\, v] \right] \right) \right\}$$

$$\equiv -A^{vm} \sum_{i,j,k} \frac{\delta_{k+1/2}[\chi]}{e_{3w}} \left\{ \delta_{k+1/2}\Big[\delta_{i+1/2}[e_{2u}\, u] + \delta_{j+1/2}[e_{1v}\, v] \Big] \right\}$$

$$\equiv -A^{vm} \sum_{i,j,k} \frac{1}{e_{3w}} \delta_{k+1/2}[\chi] \;\; \delta_{k+1/2}[e_{1T}\, e_{2T}\; \chi]$$

$$\equiv -A^{vm} \sum_{i,j,k} \frac{e_{1T}\, e_{2T}}{e_{3w}} \;\; \left(\delta_{k+1/2}[\chi] \right)^2 \qquad \equiv 0$$

C.5 Conservation Properties on Tracer Physics

The numerical schemes used for tracer subgridscale physics are written such that the heat and salt contents are conserved (equations in flux form, second order centered finite differences). Since a flux form is used to compute the temperature and salinity, the quadratic form of these quantities (i.e. their variance) globally tends to diminish. As for the advection term, there is generally no strict conservation of mass, even if in practice the mass is conserved to a very high accuracy.

C.5.1 Conservation of Tracers

constraint of conservation of tracers :

$$\int_D \nabla \cdot (A\ \nabla T)\ dv$$

$$\equiv \sum_{i,j,k} \left\{ \delta_i \left[A_u^{lT} \frac{e_{2u}\, e_{3u}}{e_{1u}} \delta_{i+1/2}\left[T\right] \right] + \delta_j \left[A_v^{lT} \frac{e_{1v}\, e_{3v}}{e_{2v}} \delta_{j+1/2}\left[T\right] \right] + \delta_k \left[A_w^{vT} \frac{e_{1T}\, e_{2T}}{e_{3T}} \delta_{k+1/2}\left[T\right] \right] \right\} \quad \equiv 0$$

In fact, this property simply results from the flux form of the operator.

C.5.2 Dissipation of Tracer Variance

constraint on the dissipation of tracer variance :

$$\int_D T\ \nabla \cdot (A\ \nabla T)\ dv$$

$$\equiv \sum_{i,j,k} T \left\{ \delta_i \left[A_u^{lT} \frac{e_{2u}\, e_{3u}}{e_{1u}} \delta_{i+1/2}\left[T\right] \right] + \delta_j \left[A_v^{lT} \frac{e_{1v}\, e_{3v}}{e_{2v}} \delta_{j+1/2}\left[T\right] \right] + \delta_k \left[A_w^{vT} \frac{e_{1T}\, e_{2T}}{e_{3T}} \delta_{k+1/2}\left[T\right] \right] \right\}$$

$$\equiv -\sum_{i,j,k} \left\{ A_u^{lT} \left(\frac{1}{e_{1u}} \delta_{i+1/2}\left[T\right] \right)^2 e_{1u}\, e_{2u}\, e_{3u} + A_v^{lT} \left(\frac{1}{e_{2v}} \delta_{j+1/2}\left[T\right] \right)^2 e_{1v}\, e_{2v}\, e_{3v} + A_w^{vT} \left(\frac{1}{e_{3w}} \delta_{k+1/2}\left[T\right] \right)^2 e_{1w}\, e_{2w}\, e_{3w} \right\} \quad \leq 0$$