NEMO ocean engine

NEMO v2.3

– Draft –

Gurvan Madec - gurvan.madec@locean-ipsl.umpc.fr

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

10 octobre 2007

Table des matières

Abstract / Résumé

The ocean engine of NEMO is a primitive equation model adapted to regional and global ocean circulation. 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, the 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 z full or partial step 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 turbulent closures for the vertical mixing. Within NEMO, the ocean is interfaced with a sea-ice model (LIM), passive tracer and biogeochemical models (TOP) and, via the OASIS coupler, with several atmospheric general circulation models.

Le moteur océanique de NEMO 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, la hauteur de la mer, 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 une fermeture turbulente TKE et KPP pour le mélange vertical. Via 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

OPA (an acronym for Ocean PArallélisé) is the ocean component of NEMO (Nucleus for European Modelling of the Ocean (www.locean-ipsl.upmc.fr/NEMO). Like all components of NEMO, it is developed under the CECILL license, which is a french adaptation of the GNU GPL (General Public license). Anyone may use OPA freely for research purposes, and is encouraged to communicate back to the NEMO team its own developments and improvements. The model and the present document have been made available as a service to the community. We cannot certify that the code and its manual are free of errors. Bugs are inevitable and some have undoubtedly survived the testing phase. Users are encouraged to bring them to our attention. The author assumes no responsibility for problems, errors, or incorrect usage of OPA.

The OPA OGCM reference in papers and other publications is as follows :

Madec, G., 2007 : NEMO ocean engine. *Note du Pôle de modélisation*, Institut Pierre-Simon Laplace (IPSL), France, NXX, YYpp .

1 Introduction

The Nucleus for European Modelling of the Ocean (*NEMO*) is a framework of ocean related engines, namely OPA for the Ocean dynamics and thermodynamics, LIM for the sea-ice dynamics and thermodynamics, TOP for the biogeochemistry (both transport (TRP) and sources minus sinks (LOBSTER, PISCES). 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. This documentation provides information about the physics represented by the OPA ocean model 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 OPA8.2 model described in Madec et al. [1998]. This model has been used for a wide range of applications, either regional or global, as a forced ocean model or 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, s, and variable volumes). Momentum equations are formulated in the vector invariant form. The model equations are written in dimensional units in the meter, kilogram, second (MKS) international system.

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. Vertical discretisation uses z-coordinates (including partial step), s- (terrain-following) coordinate (fixed volume thickness and linear free surface), or $s*$-

coordinate (variable volume thickness and nonlinear free surface). The following chapters describe the discretisation of the prognostic equations (momentum and tracers). Explicit, split-explicit or implicit free surface formulations are implemented as well as arid-lid approximation. 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).

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.

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 [Valcke 2006].

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 [Gent and Mcwilliams 1990] with a space and time variable coefficient Tréguier et al. [1997]. The model has vertical harmonic viscosity and diffusion with a space and time variable coefficient, with options to compute the coefficients with Blanke and Delecluse [1993], Large et al. [1994], or Pacanowski and Philander [1981] 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. In particular, each trend (e.g., 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 the three-letter rule. Each module name is made of three-letter sequences. For example, *tradmp.F90* is a module related to the TRAcers equation, computing the DaMPing. The complete list of module names is presented in annex. 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 follows this organization. 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 (v2.3), LBC directory (see Chap. 7) does not yet exist. When created LBC will gather OBC directory (Open Boundary Condition), *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 model include in the release 2.3 of NEMO. This release differs significantly from version 8, documented in Madec et al. [1998]. The major 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 [1],
(4) more choices for the treatment of the free surface : full explicit, split-explicit , filtered and rigid-lid
(5) non linear free surface option (variable level thickness distributed on the whole water column)
(6) additional schemes for vector and flux forms of the momentum advection
(7) addition of several advection schemes on 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.

[1] 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 true equation of state.

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

At the time of this writing, the current release is NEMO 2.3. The new surface module described in this document is not yet part of the current distribution.

Red color : not in the current reference version (v2.3) but expected soon.

Yellow color : missing references, text to be updated.

2 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 buoyancy force (this removes convective processes from the initial Navier-Stokes equations : they must be parameterized)

(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 quite 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**) plan), 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, the heat and salt conservation and an equation of state) :

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

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

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

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

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

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

where ∇ is the generalised derivative vector operator in $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ directions, t the time, z the vertical coordinate, ρ the *in situ* density given by the equation of state (2.1f), ρ_o a reference density, p the pressure, $f = 2\boldsymbol{\Omega} \cdot \mathbf{k}$ the Coriolis acceleration (where $\boldsymbol{\Omega}$ is the Earth angular velocity vector), and g the gravitational acceleration. $\mathbf{D^U}$, D^T and D^S are the parameterizations of small-scale physics for momentum, temperature and salinity, including surface forcing terms. Their nature and formulation are discussed in §2.6, page 26.

.

2.1.2 Boundary Conditions

An ocean is bounded by complex coastlines and bottom topography at its base and by 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 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 surfaces, 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.

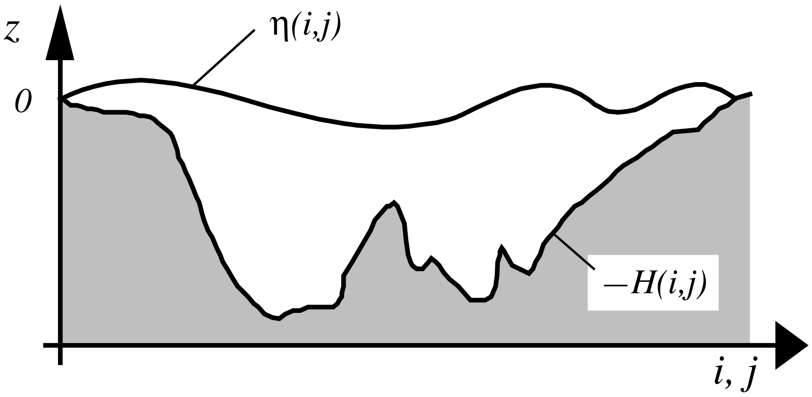

FIG. 2.1 – The ocean is bounded by two surfaces, $z = -H(i, j)$ and $z = \eta(i, j, k, t)$, where H is the depth of the sea floor and η the height of the sea surface. Both H and η are referenced to $z = 0$.

Land - ocean interface : the major flux between continental surfaces and the ocean is a mass exchange of fresh water through river runoff. Such an exchange modifies locally 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 to close the water cycle of the climatic 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 across the sea floor are negligibly small, except in special areas of little extent. They are always neglected in the model. 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 friction processes. Such momentum transfer occurs at small scales in a boundary layer. It must be parameterized in terms of turbulent fluxes through bottom and/or lateral boundary conditions. Its specification depends on the nature of the physical parameterization used for $\mathbf{D^U}$ in (2.1a). They are discussed in in §2.6.1, page 26.

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 two media 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 ($\sim\ 5\,psu$) 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.

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 an 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 :

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

The surface pressure requires a more specific treatment. Two strategies can be considered : (a) the introduction of a new variable η, the free-surface elevation, for which a prognostic equation can be established and solved ; (b) the assumption 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, a solution of the free-surface elevation consists in 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 these waves as 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 OPA, 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 kinematics 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 adequate : 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 a filter to slow down the fastest waves while not altering the slow barotropic Rossby waves. In that case it is also generally sufficient to solve a linearized version of (2.5), which allows to take into account freshwater fluxes applied at the ocean surface [Roullet and Madec 2000].

• 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 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 [Killworth et al. 1991, Zhang and Endoh 1992] or the implicit scheme [Dukowicz and Smith 1994]. In OPA, we use a slightly different approach developed by Roullet and Madec [2000] : 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 homogeneous to a time which characterizes the force, $\widetilde{\rho} = \rho/\rho_o$ is the dimensionless density, and 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, $-g$ $\mathbf{A}$ η 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 propagates, $i.e.$ 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 $T_c > \Delta t$. Roullet and Madec [2000] 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 model layers, the free surface equation (2.5) can be linearized. As emphasized by Roullet and Madec [2000] 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 in 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, Roullet and Madec [2000] 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 [Roullet and Madec 2000]. 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 spurious sources or sinks of salt [Roullet and Madec 2000].

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, can be derived from a volume transport streamfunction ψ :

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

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) :

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

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 (I.2.7 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 :

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

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 in introducing an appropriate coordinate transformation that shifts the singular point on land [Madec and Imbard 1996, Murray 1996]. As a conclusion, 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, with conservation of the local vertical. Here we give the simplified equations for this particular case. The general case is detailed by Eiseman and Stone [1980] 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 (2.3.1). Let (λ, φ, z) be the geographical coordinates 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 (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 :

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

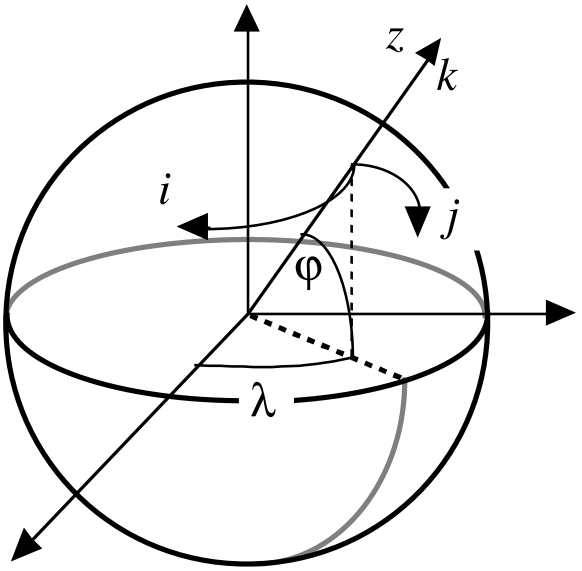

FIG. 2.2 – the geographical coordinate system (λ, φ, z) and the curvilinear coordinate system (**i**,**j**,**k**).

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 :

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

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

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

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

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

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 $\mathbf{U} = (\mathrm{u}, \mathrm{v}, \mathrm{w}) = \mathbf{U}_\mathrm{h} + \mathrm{w}\ \mathbf{k}$, 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 :

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

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

Using the fact that 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. 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 :

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

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

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

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

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

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

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

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

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

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

The flux form has two terms, the first one is expressed as the divergence of momentum fluxes (so 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 :

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

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 $f\to f+(u/a)\tan\varphi$.

The equations solved by the ocean model can be written in the following tensorial formalism :

- vector invariant form of the momentum equations :

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

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

where ζ is given by (2.20) and the surface pressure gradient is given by :

$*$ free surface formulation

$$\frac{1}{\rho_o}\nabla_h p_s = \begin{pmatrix} \frac{g}{e_1}\frac{\partial \eta}{\partial i} \\ \frac{g}{e_2}\frac{\partial \eta}{\partial j} \end{pmatrix} \qquad \text{where } \eta \text{ is solution of (2.5)} \tag{2.26}$$

$*$ rigid-lid approximation

$$\frac{1}{\rho_o}\nabla_h p_s = \begin{pmatrix} \overline{M}_u + \frac{1}{H\,e_2}\frac{\partial}{\partial j}\left(\frac{\partial \psi}{\partial t}\right) \\ \overline{M}_v - \frac{1}{H\,e_1}\frac{\partial}{\partial i}\left(\frac{\partial \psi}{\partial t}\right) \end{pmatrix} \tag{2.27}$$

where $\mathbf{M} = (M_u, M_v)$ represents the collected contributions of nonlinear, viscous 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 :

$$\frac{\partial}{\partial i}\left[\frac{e_2}{H\,e_1}\frac{\partial}{\partial i}\left(\frac{\partial \psi}{\partial t}\right)\right] + \frac{\partial}{\partial j}\left[\frac{e_1}{H\,e_2}\frac{\partial}{\partial j}\left(\frac{\partial \psi}{\partial t}\right)\right] = \frac{\partial}{\partial i}\left(e_2 \overline{M}_v\right) - \frac{\partial}{\partial j}\left(e_1 \overline{M}_u\right) \tag{2.28}$$

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

$$\frac{\partial w}{\partial k} = -\chi\, e_3 \tag{2.29}$$

$$\frac{\partial p_h}{\partial k} = -\rho\, g\, e_3 \tag{2.30}$$

where the divergence of the horizontal velocity, χ is given by (I.3.8).

- tracer equations :

$$\frac{\partial T}{\partial t} = -\frac{1}{e_1 e_2}\left[\frac{\partial\,(e_2 T\,u)}{\partial i} + \frac{\partial\,(e_1 T\,v)}{\partial j}\right] - \frac{1}{e_3}\frac{\partial\,(T\,w)}{\partial k} + D^T \tag{2.31}$$

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

$$\rho = \rho\,(T, S, z(k)) \tag{2.33}$$

The expression of $\mathbf{D}^U$, D^S andD^T depends on the subgrid scale parameterization used. It will be defined in §2.6.1.

2.4 Curvilinear *s*-coordinate System

2.4.1 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.

A terrain-following coordinate system (hereafter $s-$coordinates) avoids the discretisation error in the depth field since the layers of computation are gradually adjusted with depth to the ocean bottom. Relatively shallow topographic features 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) as can gentle, large-scale slopes of the sea floor. 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 $s-$coordinates it is possible, at least in principle, to have the bottom and the sea surface as the only boundaries of the domain. Nevertheless, $s-$coordinates also have 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 $s-$coordinates consists of two terms (see Appendix A),

$$\nabla p|_z = \nabla p|_s - \frac{\partial p}{\partial s} \nabla z|_s \tag{2.34}$$

The second term in (2.34) 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 $\sigma-$coordinate (i.e. a depth-normalised coordinate system $\sigma = z/H$), Haney [1991] and Beckmann and Haidvogel [1993] have given estimates of the magnitude of this truncation error. It depends on topographic slope, stratification, horizontal and vertical resolution, 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 $s-$coordinates and step-like representation of bottom topography [Madec et al. 1996]. However, another problem is then raised in the definition of the model domain.

Aike Beckmann's solution

A minimum of diffusion along the coordinate surfaces of any finite difference model is always required for numerical reasons. 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) [Madec et al. 1996]. An alternate solution consists in rotating the lateral diffusive tensor to geopotential or to isoneutral surfaces (see §2.6.1 and Appendix B .

2.4.2 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 $s = s(i, j, k)$, which includes $z-$ and $\sigma-$coordinates as special cases ($s = z$ and $s = \sigma = z/H$, 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)), and the slopes in the (**i,j**) directions between $s-$ and $z-$surfaces by :

$$\sigma_1 = \frac{1}{e_1} \; \left.\frac{\partial z}{\partial i}\right|_s \quad \text{, and} \quad \sigma_2 = \frac{1}{e_2} \; \left.\frac{\partial z}{\partial j}\right|_s \tag{2.35}$$

We also introduce a "vertical" velocity ω defined as the velocity normal to $s-$surfaces :

$$\omega = w - \sigma_1 \, u - \sigma_2 \, v \tag{2.36}$$

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

* momentum equation :

$$\begin{aligned}\frac{\partial u}{\partial t} = &+ (\zeta + f) \; v - \frac{1}{e_3} \omega \frac{\partial u}{\partial k} - \frac{1}{e_1} \frac{\partial}{\partial i} \left(\frac{1}{2} \left(u^2 + v^2 \right) + \frac{p_h}{\rho_o} \right) \\ &+ g \frac{\rho}{\rho_o} \sigma_1 - \frac{1}{\rho_o e_1} \frac{\partial p_s}{\partial i} + D_u^{\mathbf{U}}\end{aligned} \tag{2.37}$$

$$\begin{split}\frac{\partial v}{\partial t} = & -\left(\zeta + f\right) u - \frac{1}{e_3}\omega \frac{\partial v}{\partial k} - \frac{1}{e_2}\frac{\partial}{\partial j}\left(\frac{1}{2}\left(u^2+v^2\right)+\frac{p_h}{\rho_o}\right) \\ & + g\frac{\rho}{\rho_o}\sigma_2 - \frac{1}{\rho_o e_2}\frac{\partial p_s}{\partial j} + D_v^{\mathbf{U}}\end{split} \quad (2.38)$$

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 same equation as w, i.e. (2.29), with χ, the divergence of the horizontal velocity field given by :

$$\chi = \frac{1}{e_1 e_2 e_3}\left[\frac{\partial\left(e_2 e_3\, u\right)}{\partial i} + \frac{\partial\left(e_1 e_3\, v\right)}{\partial j}\right] \quad (2.39)$$

* tracer equations :

$$\frac{\partial T}{\partial t} = -\frac{1}{e_1 e_2 e_3}\left[\frac{\partial\left(e_2 e_3 T\, u\right)}{\partial i} + \frac{\partial\left(e_1 e_3 T\, v\right)}{\partial j}\right] - \frac{1}{e_3}\frac{\partial\left(T\,\omega\right)}{\partial k} + D^T \quad (2.40)$$

$$\frac{\partial S}{\partial t} = -\frac{1}{e_1 e_2 e_3}\left[\frac{\partial\left(e_2 e_3 S\, u\right)}{\partial i} + \frac{\partial\left(e_1 e_3 S\, v\right)}{\partial j}\right] - \frac{1}{e_3}\frac{\partial\left(S\,\omega\right)}{\partial k} + D^S \quad (2.41)$$

The equation of state has the same expression as in $z-$coordinates. The expression of $\mathbf{D}^U$, D^S andD^T depends on the subgrid scale parameterization used. It will be defined in §2.6.

to be updated ==> The whole set of the continuous equations solved by the model in the $s-$coordinate system is summarised in Table I.2.

Add a few works on z and zps and s and underlies the differences between all of them <== end update

2.5 Curvilinear z*- or s* coordinate System

to be updated ==>

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 [Levier et al. 2007] available on the *NEMO*web site.

<== end update

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 occurs (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 :

$$\begin{aligned} \mathbf{D}^{v\mathbf{U}} &= \frac{\partial}{\partial z}\left(A^{vm}\frac{\partial \mathbf{U}_h}{\partial z}\right) \ , \\ D^{vT} &= \frac{\partial}{\partial z}\left(A^{vT}\frac{\partial T}{\partial z}\right) \ , \quad D^{vS} = \frac{\partial}{\partial z}\left(A^{vT}\frac{\partial S}{\partial z}\right) \end{aligned} \tag{2.42}$$

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). 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 (as Richardson number, Brunt-Vaisälä frequency...), or computed from a turbulent closure model. The choices available in OPA are discussed in §9).

2.6.2 Lateral Diffusive and Viscous Operators Formulation

Lateral turbulence can be roughly divided into a mesoscale turbulence associated to eddies which can be solved explicitly if the resolution is sufficient as 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 isoneutral surfaces (or more precisely neutral surfaces, i.e. isoneutral surfaces referenced at the local depth) rather than across them. As the slope of isoneutral 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 isoneutral 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). Gent and Mcwilliams [1990] have proposed a parameterization 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 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.

All these parameterizations of subgrid scale physics present advantages and disadvantages. There are not all available in OPA. In the $z-$coordinate formulation, four options are offered for active tracers (temperature and salinity) : second order geopotential operator, second order isoneutral operator, Gent and Mcwilliams [1990] parameterization and fourth order geopotential operator. The same options are available for momentum, except Gent and Mcwilliams [1990] parameterization which only involves tracers. In $s-$coordinate formulation, an additional option is 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) :

$$D^{lT} = \nabla . \left(A^{lT} \; \Re \; \nabla T \right) \qquad \text{with} \qquad \Re = \begin{pmatrix} 1 & 0 & -r_1 \\ 0 & 1 & -r_2 \\ -r_1 & -r_2 & r_1^2 + r_2^2 \end{pmatrix} \tag{2.43}$$

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), and r_1 and r_2 is the differential operator defined in §2.3 or §2.4 depending on the vertical coordinate used. Note that the formulation of r_1 and r_2 is exact for the slopes between geopotential and $s-$surfaces, while it is only an approximation for the slopes between isoneutral and z or $s-$surfaces. Indeed, in the latter case, two assumptions are made to simplify r_1 and r_2 [Cox 1987] : the ratio between lateral and vertical diffusive coefficients is known to be several orders of magnitude smaller than unity, and the slopes are, generally less than 10^2 in the ocean (see Appendix B). This leads to the linear tensor (2.43) where the two isoneutral directions of diffusion are independent and where the diapycnal diffusivity contribution is solely along the vertical.

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 and r_2) while in $s-$coordinate they are equal to σ_1 and σ_2, respectively (see (2.35)).

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 $s-$coor-dinates :

$$r_1 = \frac{e_3}{e_1} \left(\frac{\partial \rho}{\partial i} \right) \left(\frac{\partial \rho}{\partial k} \right)^{-1} \quad , \quad r_1 = \frac{e_3}{e_1} \left(\frac{\partial \rho}{\partial i} \right) \left(\frac{\partial \rho}{\partial k} \right)^{-1} \tag{2.44}$$

When the *Eddy Induced Velocity* parametrisation (eiv) [Gent and Mcwilliams 1990] is used, an additional tracer advection is introduced in combination with the isoneutral diffusion of tracers :

$$D^{lT} = \nabla \cdot \left(A^{lT} \; \Re \; \nabla T \right) + \nabla \cdot \left(\mathbf{U}^* \, T \right) \tag{2.45}$$

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

$$\begin{aligned} u^* &= +\frac{1}{e_3} \frac{\partial}{\partial k} \left[A^{eiv} \; \tilde{r}_1 \right] \\ v^* &= +\frac{1}{e_3} \frac{\partial}{\partial k} \left[A^{eiv} \; \tilde{r}_2 \right] \\ w^* &= -\frac{1}{e_1 e_2} \left[\frac{\partial}{\partial i} \left(A^{eiv} \; e_2 \, \tilde{r}_1 \right) + \frac{\partial}{\partial j} \left(A^{eiv} \; e_1 \, \tilde{r}_2 \right) \right] \end{aligned} \tag{2.46}$$

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 *geopo-*

tential surfaces and thus depends on the coordinate considered :

$$\tilde{r}_n = \begin{cases} r_n & \text{in } z\text{−coordinate} \\ r_n + \sigma_n & \text{in } s\text{−coordinate} \end{cases} \quad \text{where } n = 1, 2 \tag{2.47}$$

The normal component of the eddy induced velocity is zero at all the boundaries. this can be achieved by tapering either the eddy coefficient or the slopes to zero in the vicinity of the boundaries.

lateral fourth order tracer diffusive operator

The lateral fourth order tracer diffusive operator is defined by :

$$D^{lT} = \Delta \left(A^{lT} \; \Delta T \right) \qquad \text{where } D^{lT} = \Delta \left(A^{lT} \; \Delta T \right) \tag{2.48}$$

It is the second order operator given by (2.43) 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 (2.19c) to the horizontal velocity vector (see Appendix B) :

$$\begin{aligned} \mathbf{D}^{l\mathbf{U}} = &\quad \nabla_h \left(A^{lm} \chi \right) \; - \; \nabla_h \times \left(A^{lm} \, \zeta \; \mathbf{k} \right) \\ = &\left(\begin{array}{c} \dfrac{1}{e_1} \dfrac{\partial \left(A^{lm} \chi \right)}{\partial i} - \dfrac{1}{e_2 e_3} \dfrac{\partial \left(A^{lm} \; e_3 \zeta \right)}{\partial j} \\ \dfrac{1}{e_2} \dfrac{\partial \left(A^{lm} \chi \right)}{\partial j} + \dfrac{1}{e_1 e_3} \dfrac{\partial \left(A^{lm} \; e_3 \zeta \right)}{\partial i} \end{array} \right) \end{aligned} \tag{2.49}$$

Such a formulation ensures a complete separation between the vorticity and horizontal divergence fields (§ II.4c). Unfortunately, it is not available for geopotential diffusion in $s-$coordinates and for isoneutral diffusion. In these two cases, the u and $v-$fields are considered as independent scalar fields, so that the diffusive operator is given by :

$$\begin{aligned} D_u^{l\mathbf{U}} &= \nabla . \left(\Re \; \nabla u \right) \\ D_v^{l\mathbf{U}} &= \nabla . \left(\Re \; \nabla v \right) \end{aligned} \tag{2.50}$$

where $\Re$ is given by (I.5.2). It is the same expression as those used for diffusive operator on tracers.

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 (2.49) or (2.49) with the eddy viscosity coefficient correctly placed :

geopotential diffusion in $z-$coordinates :

$$\begin{aligned} \mathbf{D}^{l\mathbf{U}} &= \nabla_h \left\{ \; \nabla_h \cdot \left[A^{lm} \, \nabla_h \left(\chi \right) \right] \; \right\} \\ & \quad + \nabla_h \times \left\{ \; \mathbf{k} \cdot \nabla \times \left[A^{lm} \, \nabla_h \times \left(\zeta \; \mathbf{k} \right) \right] \; \right\} \end{aligned} \tag{2.51}$$

geopotential diffusion in $s-$coordinates :

$$\left\{ \begin{aligned} D_u^{l\mathbf{U}} &= \Delta \left(A^{lm} \; \Delta u \right) \\ D_v^{l\mathbf{U}} &= \Delta \left(A^{lm} \; \Delta v \right) \end{aligned} \right. \quad \text{where} \quad \Delta \left(\bullet \right) = \nabla \cdot \left(\Re \nabla (\bullet) \right) \tag{2.52}$$

3 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 OPA, 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 in 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 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 allows satisfying many of the continuous properties (see Annexe 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).

3.1.2 Discrete Operators

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

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

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

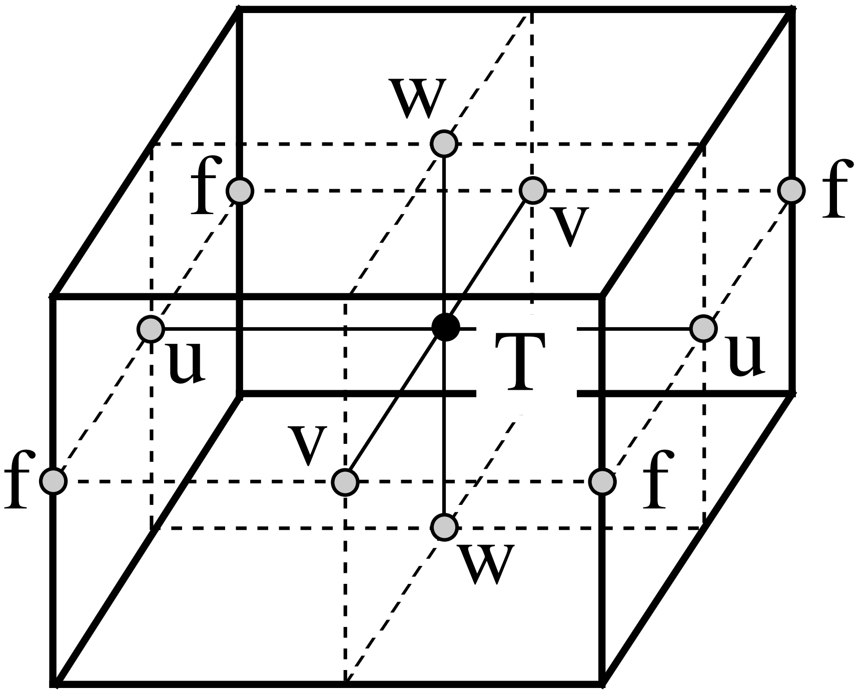

FIG. 3.1 – Arrangement of variables. T indicates scalar points where temperature, salinity, density, pressure and horizontal divergence are defined. $(u$,v,$w)$ indicates vector points, and f indicates vorticity points where both relative and planetary vorticities are defined

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 T-point has its three components defined at (u,v,w) while its Laplacien is defined at T-point. These operators have the following discrete forms in the curvilinear s-coordinate system :

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

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 indexation is only used for the writing of semi-discrete equation. In the code, the indexation use integer value only and has a reverse direction in the vertical (see §3.1.3)

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

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, f) and its divergence defined at T-points :

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

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

In the special case of pure z-coordinates 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. It can be simplified from outside and inside the δ_i and δ_j operators.

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

In continuous, 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 straight forward 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 $\mathbf{A}$ has its components defined at vector points (u, v, w).

Let a and b be two fields defined on the ocean mesh, extended to zero inside continental area. By integration by part it can be shown that the derivation operators (δ_i, δ_j and δ_k) are anti-symmetric linear operators, and further that the averaging operators $\overline{\cdot}^{\,i}$, $\overline{\cdot}^{\,j}$ 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 derivation 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 Indexation

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

Horizontal Indexation

The indexation 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 nearby northeast f-point have the same i-and j-indices.

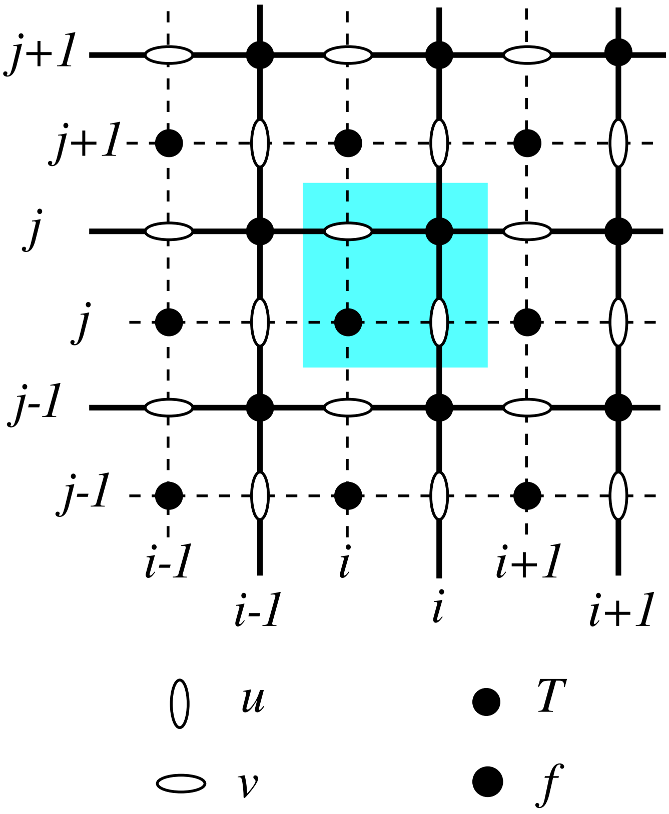

FIG. 3.2 – Horizontal integer indexation used in the FORTRAN code. The dashed area indicates the cell in which variables contained in arrays have the same i- and j-indices

Vertical Indexation

In the vertical plane, the chosen indexation requires special attention since the k-axis is re-oriented downward in the FORTRAN code compared to the indexation used for the semi-discrete equations and given in §3.1.1. The sea surface corresponds to the w-level $k = 1$ like the $T-$level just below (Fig.3.1.3). The last w-level ($k = jpk$) is either the ocean bottom or inside the ocean floor while the last $T-$level is always inside the floor (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 nearby 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). As the scale factors

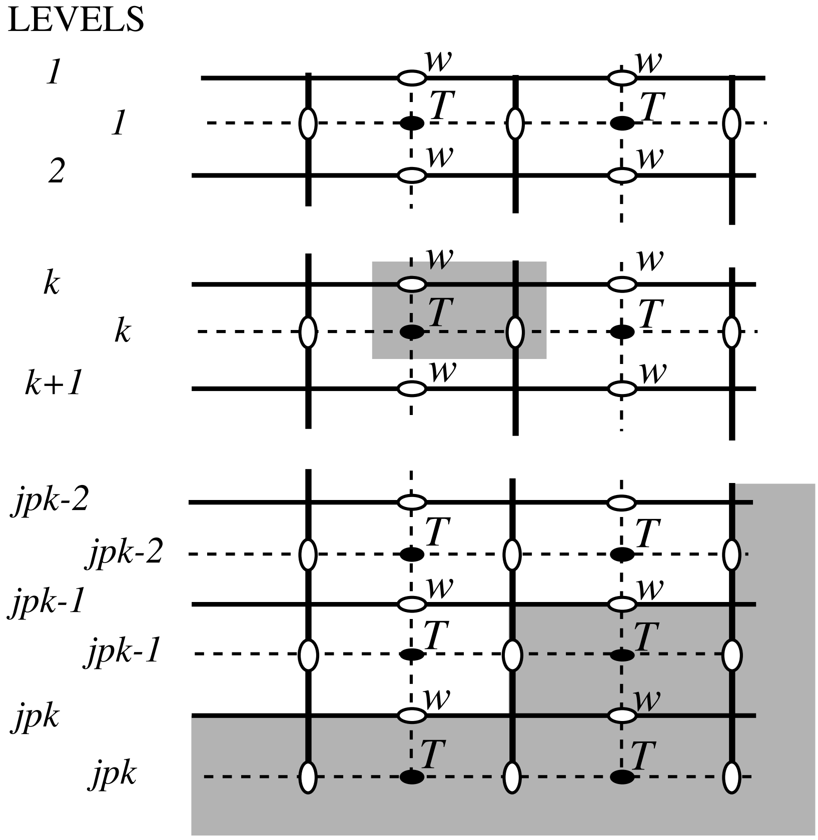

FIG. 3.3 – Vertical integer indexation used in the FORTRAN code. Note that the k-axis is oriented downward. The dashed area indicates the cell in which variables contained in arrays have the same k-index.

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 (or additional files included in this module such as *par_ORCA_R2.h90*, specific to a given configuration). The use of parameters rather than variables (together with dynamic allocation of arrays) was made 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 (ref agrif !+ ref part of the doc) . 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 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 latitude φ and the longitude λ as a function of (i, j). The horizontal scale factors are calculated using (2.18). For example, when the latitude and longitude are function of a single value (j and i, respectively) (geographical configuration of the mesh), the horizontal mesh definition reduces to define the wanted $\varphi(j)$, $\varphi'(j)$, $\lambda(i)$, and $\lambda'(i)$ 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 and scale factors of w-points are exactly equal to those of $T-$points, thus no specific arrays are defined at those grid-points.

Note that the definition of the scale factors — as the analytical first derivative of the transformation that gives (λ, φ, z) as a function of (i, j, k) — is specific to the OPA model [Marti et al. 1992]. 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 smooth grids [Tréguier et al. 1996]. An example of the effect of such a choice is shown in Fig. 3.2.1.

3.2.2 Choice of horizontal grids

The user has three options to define a horizontal grid, involving the parameter $jphgr\_mesh$ of the *par_oce.F90* module.

1. For 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 : *jphgr_mesh*=0.
2. A few simple analytical grids are provided as examples, that can be selected by setting *jphgr_mesh*=1 to 5 (see below)
3. 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 is regular in space, with grid sizes specified by parameters *ppe1_deg* and *ppe2_deg*, respectively. 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 that 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 those two cases (*jphgr_mesh*=1 or 4), the grid position is defined by the longitude and latitude of the south-westhernmost point (*ppglamt0* and *ppgphi0*). Note that for the Mercator grid the user need only provide an approximate starting latitude : the real

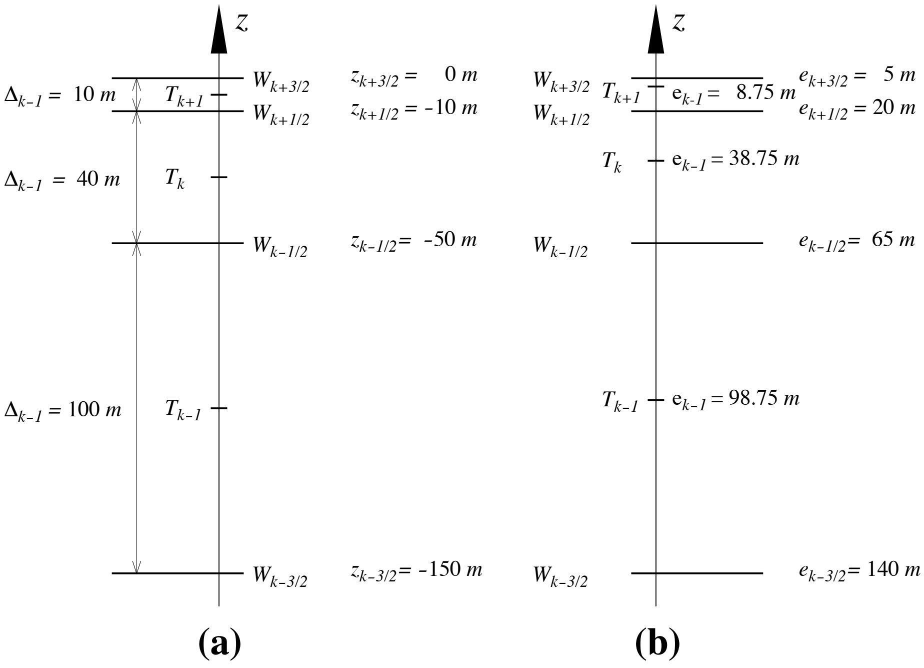

FIG. 3.4 – (a) Traditional definition of grid-point position and grid-size in the vertical versus (b) analytically derived grid-point position and scale factors. For both grid here,a same w-point depth has been chosen but in (a) the T-points are set at the middle of w-points while in (b) they are defined 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.

latitude will be recalculated analytically, so as to ensure that the equator corresponds to a T- and$U$-point.

Rectangular grids ignoring the spherical geometry are defined with *jphgr_mesh* = 2, 3, 5. The domain is either a 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 each direction, and given in meters 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 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 related 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 the insufficient model resolution). On example is Lombok Strait in the ORCA2 configuration. When such modifications are done, the output grid written when *nmsh* $\neq 0$ is not exactly equal to the input grid.

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

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

```
!-----------------------------------------------------------------------
&namdom    !   space and time domain (bathymetry, mesh, timestep)
!-----------------------------------------------------------------------
   ntopo      =     1       !  = 1 read the bathymetry_level
   e3zps_min =      5.      !  minimum thickness of the partial step is the min of
   e3zps_rat =      0.1     !  e3zps_min and e3zps_rat * e3t  (with 0<e3zps_rat<1)
   nmsh       =     0       !  =1 create a mesh file (coordinates, scale factors, masks)
   nacc       =     0       !  the acceleration of convergence method
   !                        !     = 0, no acceleration, rdt = rdttra
   !                        !     = 1, acceleration used, rdt < rdttra(k)
   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
   rdtmax     =  5760.      !  maximum time step on tracers
   rdth       =   800.      !  depth variation of tracer time step
   rdtbt      =    90.      !  barotropic time step (for the time splitting algorithm)
   nfice      =     5       !  frequency of ice model call
   nfbulk     =     5       !  frequency of bulk formulea call (not used if ice used)
   nclosea    =     0       !  = 0 no closed sea in the model domain
!                           !  = 1 closed 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).

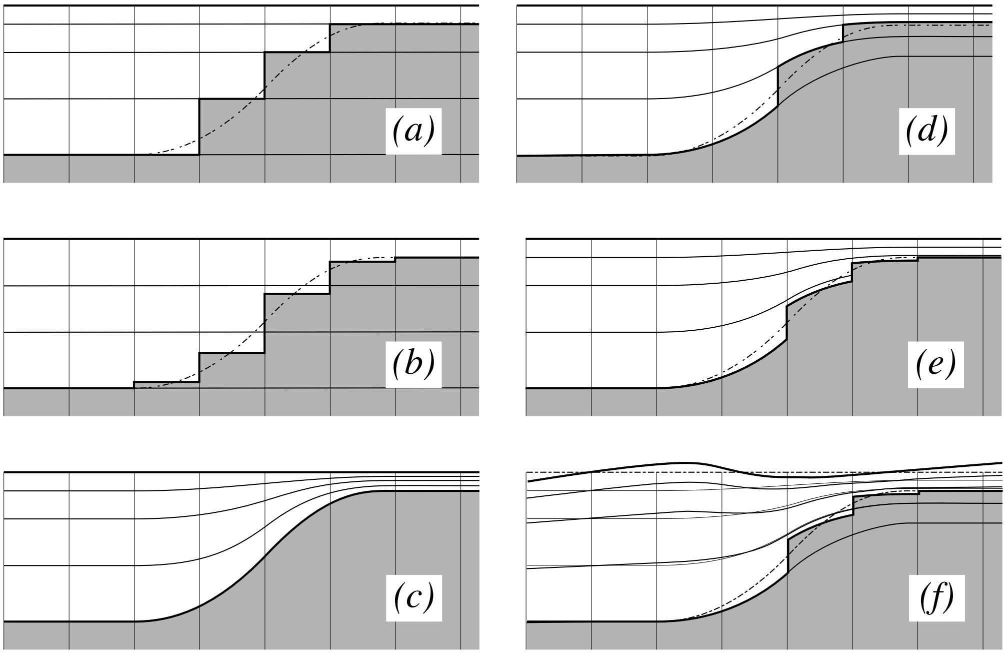

FIG. 3.5 – The ocean bottom as seen by the model : (a) z-coordinate with full step, (b) z-coordinate with partial step, (c) s-coordinate : terrain following representation, (d) hybrid $s-z$ coordinate, (e) hybrid $s-z$ coordinate with partial step, and (f) same as (e) but with variable volume level associated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

The choice of a vertical coordinate among all those offered in NEMO, 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 : hybrid $s-z$ or $s-zps$ coordinate (Fig. 3.3d and 3.3e). When using the variable volume option **key_vvl**), 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-coordinates (hybride and partial step coordinates not yet implemented 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). 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 height at the deepest levels using the bathymetry, to obtain the final three-dimensional depth and scale factor arrays.

sco Smooth the bathymetry to fullfill the hydrostatic consistency criteria and set the three-dimensional transformation.

s-z* and *s-zps Smooth the bathymetry to fullfill 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 coordinates and scale factors are defined as functions of (i,j,k) with "fs" as prefix (examples : *fsdeptht, fse3t,* etc) that can be equal to 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 through 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 a closed basin or a periodic channel according to the parameter *jperio*.

***ntopo* = -1** a domain with a bump of topography at the central latitude and 1/3 of the domain width. 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 usually built by interpolating a standard bathymetry product (e.g., ETOPO2) onto the horizontal ocean mesh. The bathymetry file 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** defined) isolated land masses (islands) must be identified by negative integers in the input bathymetry file (see §10.7.4).

When the 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 to represent their communication with the rest of the ocean. This is unnecessary when the ocean is forced by fixed atmospheric conditions. A possibility is offered to the user to set to zero the bathymetry in rectangular regions covering those closed seas (see §10.2)

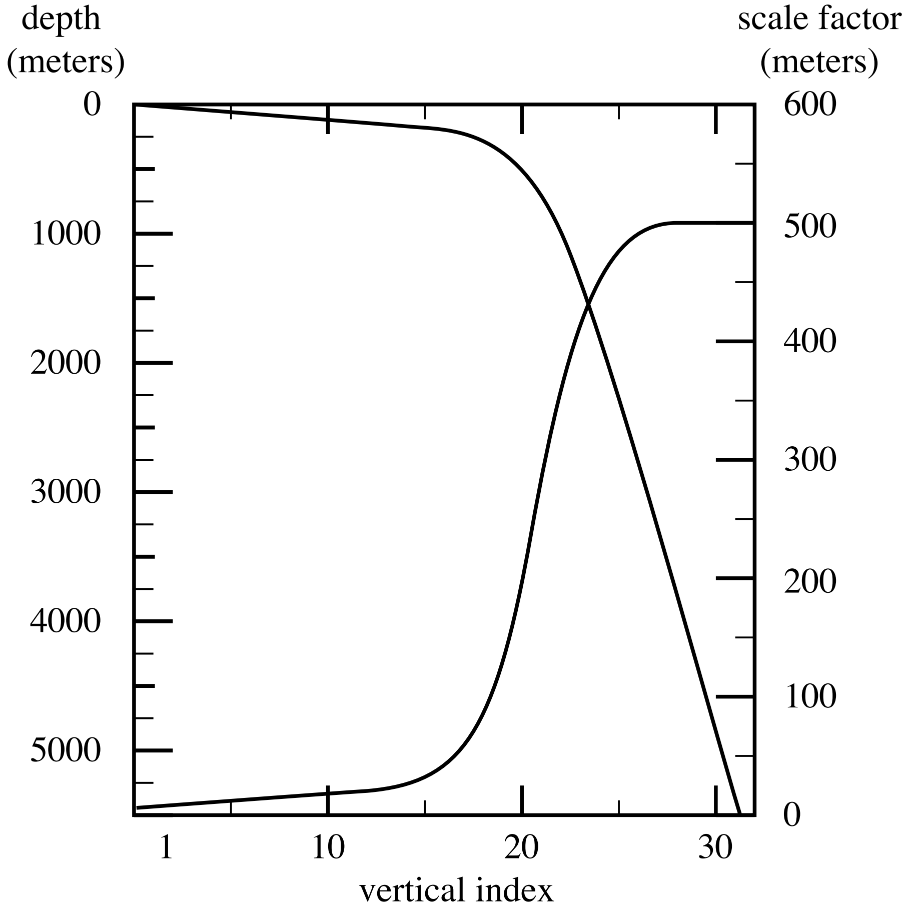

FIG. 3.6 – Default vertical mesh for ORCA2-L30. Vertical level functions for (a) T-point depth and (b) the associated scale factor as computed from (III.2.1) in z-coordinates.

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

The reference coordinate transformation $z_0(k)$ defines the arrays *gdept0* and *gdepw0* for T- and w-points, respectively. As indicated on Fig.3.1.3 *jpk* is the number of w-levels. $gdepw(1)$ being the ocean surface. There are at most *jpk*-1 T-points in 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 analytic 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 z-coordinates and partial steps :

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

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 first grid defined for ORCA2 had $10\ m$ $(500\ m)$ resolution in the surface (bottom) layers and a depth which varies from 0 at the sea surface to a minimum of $-5000\ m$. 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 vertial 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 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*=T)

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)

```
!-----------------------------------------------------------------------
&namdom    !   space and time domain (bathymetry, mesh, timestep)
!-----------------------------------------------------------------------
   ntopo      =     1       !  = 1 read the bathymetry_level
   e3zps_min =     5.       !  minimum thickness of the partial step is the min of
   e3zps_rat =     0.1      !  e3zps_min and e3zps_rat * e3t  (with 0<e3zps_rat<1)
   nmsh       =     0       !  =1 create a mesh file (coordinates, scale factors, masks)
   nacc       =     0       !  the acceleration of convergence method
   !                        !     = 0, no acceleration, rdt = rdttra
   !                        !     = 1, acceleration used, rdt < rdttra(k)
   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
   rdtmax     =  5760.      !  maximum time step on tracers
   rdth       =   800.      !  depth variation of tracer time step
   rdtbt      =    90.      !  barotropic time step (for the time splitting algorithm)
   nfice      =     5       !  frequency of ice model call
   nfbulk     =     5       !  frequency of bulk formulea call (not used if ice used)
   nclosea    =     0       !  = 0 no closed sea in the model domain
!                           !  = 1 closed sea (Caspian Sea, Great US Lakes...)
/
```

In that case, the depths of the model levels are still defined by the reference analytical function $z_0(k)$ as described in the previous section, excepted 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 layers 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 $e_{3t}(jk)$).

Add a figure here of pstep especially at last ocean level

3.3.4 s-coordinate (*ln_sco*=T)

```
!-----------------------------------------------------------------------
&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 (>0) (m)
   !                        !  (= maximum ocean depth allowed)
   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)
/
```

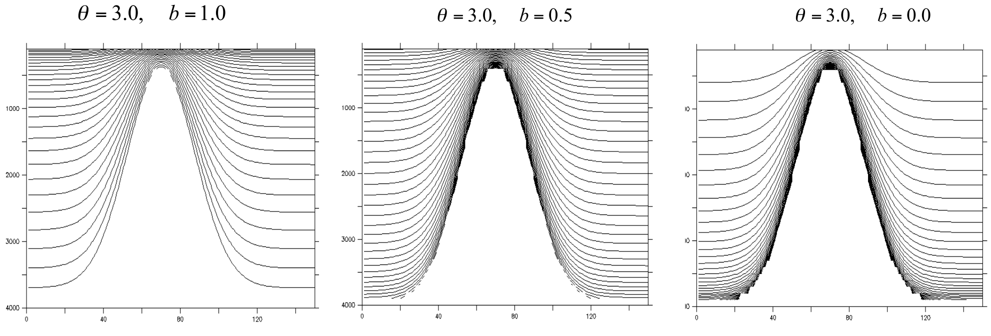

FIG. 3.7 – examples of the stretching function applied to a sea mont : from left to right, surface, surface and bottom, and bottom intensified resolution

In s-coordinate (**key_sco** defined), the depths of the model levels are defined from the product of a depth field and a stretching function and 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 $T-$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 as 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 Song and Haidvogel [1994] is provided as an example :

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

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 $0 \leqslant b \leqslant 1$. Examples of the stretching function applied to a seamount are given in 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) [Madec et al. 1996]. 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 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 case of rigid-lid approximation and islands in the computational domain (*ln_dynspg_rl*=true and **key_island** 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 of the n^{th} island ; $mbathy(i,j) = 0$, $T-$points are land points of the main land (continent) ; $mbathy(i,j) = k$, the first k T- and w-points are ocean points, the others land points. This is used to compute the island barotropic stream function used in rigid lid computation (see §**??**).

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

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

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

Add one word on tricky trick ! mbathy in further modified in zdfbfr. . . .

3.4 Time Discretisation

The time stepping used in OPA is a three level scheme that can be presented as follows :

$$x^{t+\Delta t} = x^{t-\Delta t} + 2\,\Delta t\ \mathrm{RHS}_x^{t-\Delta t, t, t+\Delta t} \qquad (3.15)$$

where x stand 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 overscripts indicate the time at which a quantity is evaluated. Each term of the RHS is evaluated at specific time step(s) depending on the physics to which it is associated. The choice of the time step used for this evaluation is discussed below as well as the implication in term of starting or restarting a model simulation. Note that the time stepping is generally performed in a one step operation : it would be dangerous to let a prognostic variable evolve in time for each term successively.

The three level scheme requires three arrays for the 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 x_a at the next time step ; 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, excepted 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 are 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" [Haltiner and Williams 1980] that develops when the method is used to model non linear fluid dynamics : the even and odd time steps tend to diverge between a physical and a computational mode. Time splitting can be controlled through the use of an Asselin time filter (first designed by [Robert 1966] and more comprehensively studied by Asselin [1972]) or by periodically reinitialising the leapfrog solution through a single integration step with a two-level scheme. In OPA 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. Recommanded values in idealized studies of shallow water turbulence are two order of magnitude lower ([Farge 1987]). Both strategies do, nevertheless, degrade the accuracy of the calculation from second to first order. The leapfrog scheme associated to a Robert-Asselin time filter has been preferred to other time differencing schemes such as predictor corrector or trapezoidal schemes because the user can better control the magnitude and the spatial structure of the time diffusion of the scheme. In association with the centred space discretisation of the advective terms in the momentum and tracer equations, it avoids implicit numerical diffusion in both the time and space discretisation 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 is unsuitable for the representation of diffusive and damping processes. For D, a horizontal diffusive terms and/or the restoring terms to a tracer climatology (when they are present, see § 4.6), a forward time differencing scheme is used :

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

This is diffusive in time and conditionally stable. For example, the condition of stability for a second and fourth order horizontal diffusions are [Griffies 2004] :

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

where e is the smallest grid size in the two horizontal direction and A^h 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 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 OPA to overcome the stability constraint : (a) a forward time differencing scheme using a time splitting technique (*ln_zdfexp*=T) or (b) a backward (or implicit) time differencing scheme by *ln_zdfexp*=F). 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 :

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

where RHS is the right hand side of the equation except 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

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

(3.22) is a linear system of equations. All the elements of the corresponding matrix vanish except those on the diagonals. 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 Richtmyer and Morton [1967]).

3.4.3 Start/Restart strategy

```
!-----------------------------------------------------------------------
&namrun   !   parameters of the run
!-----------------------------------------------------------------------
   no         =        0    !  job number
   cexper     =  "ORCA2"    !  experience name for vairmer format
   ln_rstart  = .false.     !  boolean term for restart (true or false)
   nrstdt     =        0    !  restart control = 0 restart, do not control nit000 in the restart fil
                            !                  = 1 restart, control nit000 in the restart file. Do
                            !                      use the date in the restart file (use ndate0 in
                            !                  = 2 restart, control nit000 in the restart file, use
                            !                      in the restart file. ndate0 in the namelist is ig
   nit000     =        1    !  number of the first time step
   nitend     =     5475    !  number of the first time step
   ndate0     =   010101    !  initial calendar date aammjj
   nleapy     =        0    !  Leap year calendar (0/1)
   ninist     =        0    !  initial state output flag (0/1)
   nstock     =     5475    !  number of the first time step
   nwrite     =     5475    !  frequency of OUTPUT file
   nrunoff    =        2    !  = 0 no, 1 runoff, 2 runoff+river mouth ups adv
/
```

The first time step of this three level scheme when starting from initial conditions is a forward step (Euler time integration) : $x^1 = x^0 + \Delta t\; \text{RHS}^0$.

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 double precision.

3.4.4 Time domain

add here a few word on nit000 and nitend

Write documentation on the calendar and the key variable adatrj

4 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 equation 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 : Solar Radiation penetration, and BBC : Bottom Boundary Condition), the contribution from the bottom boundary Layer (BBL) parametrisation, and an internal damping (DMP) term. The last four have been put inside brackets as they are optional. The external forcings and parameterizations require complex inputs and calculations (bulk formulae, estimation of mixing coefficients) that are carried out in modules of the SBC, LDF and ZDF categories 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 (**key_trdtra** 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
   ln_traadv_ubs    =  .false.   !  UBS scheme
/
```

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

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

which, in pure z-coordinate (**key_zco** defined), reduces to :

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

as the vertical scale factors are function of k only, and thus $e_{3u} = e_{3v} = e_{3T}$.

The flux form requires implicitly the use of the continuity equation : $\nabla \cdot (\mathbf{U}\,\mathrm{T}) = \mathbf{U} \cdot \nabla \mathrm{T}$ using $\nabla \cdot \mathbf{U} = 0$) or $\partial_t e_3 + \nabla \cdot \mathbf{U} = 0$ in variable volume case ($i.e.$ **key_vvl** defined). 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 conservation properties of the continuous equations. In other words, by substituting τ by 1 in (4.1) we recover discrete form of the continuity equation which is used to calculate the vertical velocity. The advection schemes used in OPA differ by the choice made in space and time interpolation to define the value of the tracer at the velocity points (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 : (1) in rigid-lid formulation, $w = 0$ at the surface, so the advective fluxes through the surface is zero ; (2) in non-linear free surface (variable volume case, **key_vvl** defined), convergence/divergence in the first ocean level moves up/down the free surface : there is no tracer advection through it so that the advective fluxes through the surface is also zero ; (3) in the 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 : $\tau_w|_{k=1/2} = T_{k=1}$. This boundary condition retains local conservation of tracer. Strict global conservation is not possible in linear free surface but 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 Roullet and Madec [2000], Griffies et al. [2001], Campin et al. [2004]).

The velocity field that appears in (4.1) and (4.2) is the centred (*now*) *eulerian* ocean velocity (see §5). Nevertheless, when advective bottom boundary layer (*bbl*) and/or eddy

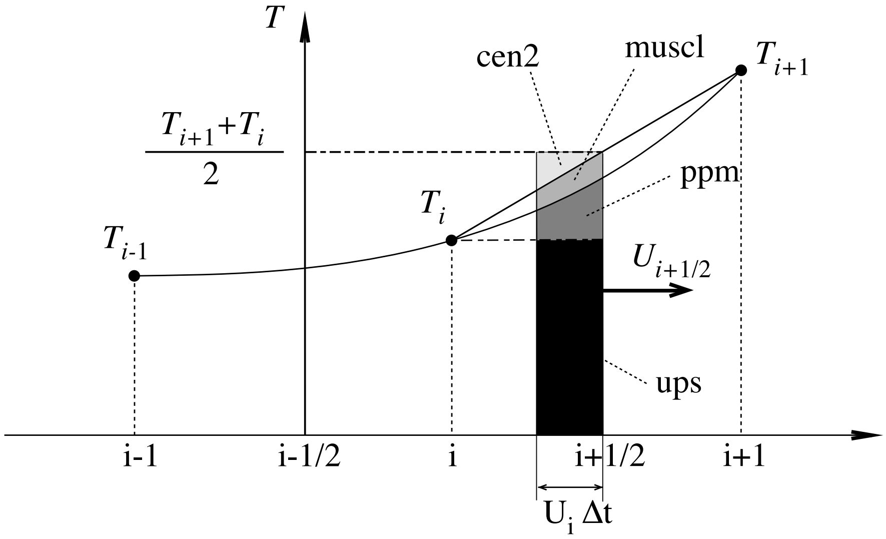

FIG. 4.1 – Schematic representation of some ways used to evaluate the tracer value at u-point and the amount of tracer exchanged between two neighbouring grid points. Upsteam biased scheme (ups) : the upstream value is used and the black area is exchanged. Piecewise parabolic method (ppm) : a parabolic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

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 *traadv_xxx.F90* module, where *xxx* is a 3 or 4 letter acronym accounting for 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, meaning false extrema are permitted. It is not recommended to use them on passive tracers ; (3) It is highly recommended to use the same advection-diffusion scheme on both active and passive tracers. In particular, if a source or sink of a passive tracer depends on a 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*=T)

In the centred second order formulation, the tracer at velocity points is evaluated as the mean of the two neighbouring T-points. 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.

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

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

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

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

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 “4^{th} order scheme” denomination 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.

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 through *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 T-grid cell abutted 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 the reduction to a second order scheme for near boundary grid point.

4.1.3 Total Variance Dissipation scheme (TVD) (*ln_traadv_tvd*=T)

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

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

where c_u is a flux limiter function taking values between 0 and 1. There exists many ways to define c_u., each correcponding to a different total variance decreasing scheme. The one chosen in OPA is described in Zalesak [1979]. 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 Lévy et al. [2001] ; note that in this paper it is referred to as "FCT" (Flux corrected transport) rather than TVD.

For stability reasons in (4.5) τ_u^{cen2} is evaluated using the *now* velocity (leap-frog environment : centred in time) while τ_u^{ups} is evaluated using the *before* velocity (diffusive part : forward in time).

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 Lévy et al. [2001]. 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 :

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

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 abutted to land and where the ocean velocity is toward land, two choices are available : use of an upstream flux (*ln_traadv_muscl*=T) or use of second order flux (*ln_traadv_muscl2*=T). Note that the latter choice does not insure the *positive* character of the scheme. Only the former can be used on both active and passive tracers.

4.1.5 Upstream Biased Scheme (UBS) (*ln_traadv_ubs*=T)

The UBS advection scheme is an upstream biased third order scheme based on an upstream-biased parabolic interpolation. It is also known as 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 [Shchepetkin and McWilliams 2005]. The overall performance of the advection scheme is similar to that reported in Farrow and Stevens [1995]. It is a relatively good compromise between accuracy and smoothness. It is not a *positive* scheme meaning false extrema are permitted but the amplitude of such are significantly reduced over the centred second order method. Nevertheless it is not recommended to apply it 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. It has therefore been preferred to evaluate the vertical flux using the TVD scheme when *ln_traadv_ubs*=T.

For stability reasons, in (4.7), the first term 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 Webb et al. [1998] in the context of the Quick advection scheme. UBS and QUICK schemes only differ by one coefficient. Substituting 1/6 with 1/8 in (4.7) leads to the QUICK advection scheme [Webb et al. 1998]. 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 *traadv_ubs.F90* module and obtain a QUICK scheme

NB 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. This possibility have been implemented and validated in ORCA05-L301. It is not currently offered in the current reference version.

NB 2 : In a forthcoming release four options will be proposed 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 Shchepetkin and McWilliams [2005] implementation of UBS in ROMS, or *(d)* an UBS. The 3^{rd} case has dispersion properties similar to an eight-order accurate conventional scheme.

NB 3 : It is straight forward to rewrite (4.7) as follows :

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

or equivalently

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

(4.9) has several advantages. First it clearly evidence that the UBS scheme is based on the fourth order scheme to which is added an upstream biased diffusive term. Second, this emphasises that the 4^{th} order part have to be evaluated at *now* time step, not only the 2^{th} order part as stated above using (4.7) and also as it is coded in NEMO v2.3. Third, the diffusive term is in fact a biharmonic operator with a eddy coefficient with is simply proportional to the velocity : $A_u^{lm} = -\frac{1}{12}\, {e_1}_u{}^3\,|u|$. Note that the current version of NEMO uses (4.7), not (4.9).

4.1.6 QUICKEST scheme (QCK) (*ln_traadv_qck*=T)

The Quadratic Upstream Interpolation for Convective Kinematics with Estimated Streaming Terms (QUICKEST) scheme proposed by Leonard [1979] is the third order Godunov scheme. It is associated with ULTIMATE QUICKEST limiter [Leonard 1991]. It has been implemented in NEMO by G. Reffray (MERCATOR-ocean).

The resulting scheme is quite expensive but *positive*. It can be used on both active and passive tracers.

4.1.7 Piecewise Parabolic Method (PPM) (*ln_traadv_ppm*=T)

The Piecewise Parabolic Method (PPM) proposed by Colella and Woodward (1984) is based on a quadradic piecewise rebuilding. As QCK scheme, it is associated with ULTIMATE QUICKEST limiter [Leonard 1991]. It has been implemented in *NEMO*by G. Reffray (MERCATOR-ocean) but is not yet offered in the current reference version.

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  (+ "key_ldfslp" if ln_sco=T)
   ln_traldf_iso    =  .true.    !     iso-neutral (+ "key_ldfslp")
                                 !  Coefficient
   !  "key_ldftra_c1d"           !     Aht = F(k)
   !  "key_ldftra_c2d"           !     Aht = F(i,j)
   !  "key_ldftra_c3d"           !     Aht = F(i,j,k)
   aht0     =  2000.             !     lateral eddy diffusivity coef. (m2/s)
   ahtb0    =     0.             !     background coef. for isopycnal diffusion (m2/s)
   aeiv0    =  2000.             !     eddy induced velocity coefficient (m2/s)
   !                             !     (+ "key_traldf_eiv")
/
```

The options available for lateral diffusion are laplacian (rotated or not) or biharmonic operators, that latter being more scale-selective (more diffusive at small scales). The specification of eddy diffusivity coefficients (either constant, variable in space and time) as well as the computation of the slope along which the operators act are performed in *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 tensor of rotation is used. This latter term is solved implicitly together with the vertical diffusion term (see §3.4)

4.2.1 Iso-level laplacian operator (*traldf_lap.F90*, *ln_traldf_lap*)

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

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

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

Notes : In pure z-coordinate (**key_zco** defined), $e_{3u} = e_{3v} = e_{3T}$, so that the vertical scale factors disappear from (4.10).

Notes : In partial step z-coordinate (*ln_zps*=T), tracers in horizontally adjacent cells are located at different depths in vicinity of the bottom. In this case, horizontal derivatives in (4.10) at the bottom level require a specific treatment. They are calculated in module zpshde, described in §4.9.

4.2.2 Rotated laplacian operator (*traldf_iso.F90*, *ln_traldf_lap*)

The general form of the second order lateral tracer subgrid scale physics (2.42) takes the following semi-discrete space form in z- and s-coordinates :

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

where r_1 and r_2 are the slopes between the surface of computation (z- or s-surfaces) and the surface along which the diffusive operator acts ($i.e.$ horizontal or iso-neutral surfaces). It is thus used when, in addition to *ln_traldf_lap*=T, we have *ln_traldf_iso*=T, or both *ln_traldf_hor*=T and *ln_zco*=T. 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 those used in the vertical physics (see §4.3). For computer efficiency reasons, this term is not computed in *traldf.F90* module, but in *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 to run safely without any additional background horizontal diffusion [Guilyardi et al. 2001]. An alternate scheme [Griffies et al. 1998] which preserves both tracer and its variance is currently been tested in *NEMO*.

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

4.2.3 Iso-level bilaplacian operator (*traldf_bilap.F90*, *ln_traldf_bilap*)

The lateral fourth order operator formulation on tracers is obtained by applying (4.10) twice. It requires an additional assumption on boundary conditions : first and third derivative terms normal to the coast are set to zero.

It is used when, in addition to *ln_traldf_bilap*=T, we have *ln_traldf_level*=T, or both *ln_traldf_hor*=T and *ln_zco*=F. In both cases, it can contributes to diapycnal mixing even if it should be less than in the laplacian case. It is therefore not recommended to use it.

Notes : In the code, the bilaplacian routine does not call twice the laplacian routine but is rather a specific routine. This is due to the fact that we introduce the eddy diffusivity coefficient, A, in the operator as : $\nabla \cdot \nabla \left(A \nabla \cdot \nabla T \right)$ and 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 (*traldf_bilapg.F90*, *ln_traldf_bilap*)

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 is used when, in addition to *ln_traldf_bilap*=T, we have *ln_traldf_iso*=T, or both *ln_traldf_hor*=T and *ln_zco*=T. 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
!-----------------------------------------------------------------------
   ln_zdfnpc = .false.        !  Non-Penetrative Convection
   avm0      = 1.2e-4         !  Kz on momemtum (m2/s)
   !                          !  (background Kz if not "key_zdfcst")
   avt0      = 1.2e-5         !  Kz for tracers (m2/s)
   !                          !  (background Kz if not "key_zdfcst")
   ln_zdfevd = .true.         !  enhanced vertical diffusion
   avevd     =   100.         !  Kz for enhanced diffusion scheme (m2/s)
   n_evdm    =     0          !  enhanced mixing Kz apply on tracer (=0)
   !                          !      or on both tracer and momentum (=1)
   ln_zdfexp =  .false.       !  =T/F  split explicit / implicit
   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, based on a laplacian operator. The vertical diffusive operator given by (2.42) 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, $A_w^{vT} = A_w^{vS}$ ecept when double diffusion mixing is parameterised (**key_zdfddm** defined). The way these coefficients can be evaluated is given in

§9 (ZDF). Furthermore, when iso-neutral mixing is used, the both mixing coefficient 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 momentum, heat and salt must be specified. At the surface they are prescribed from the surface forcing (see §4.4.1), 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 (§4.4.3).

The large eddy coefficient found in the mixed layer together with high vertical resolution implies a too restrictive constraint on the time step in explicit time stepping case (*ln_zdfexp*=True). Therefore, the default implicit time stepping is generally preferred for the vertical diffusion as it overcomes the stability constraint. A forward time differencing scheme (*ln_zdfexp*=T) 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 Roullet and Madec [2000] the forcing on an ocean tracer, c, can be split into two parts : F_{ext}^C, the flux of tracer crossing the sea surface and not linked with the water exchange d at the surface with the atmosphere, and F_{wf}^C the forcing on the concentration associated with the water flux. The latter forcing has also 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** 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, excepted 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 precipitations 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 in the water column), the product 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 (excepted 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 trasbc. Because of the hypothesis made for the temperature of precipitation and runoffs, for temperature $F^e_{wf} + F^i_{wf} = 0$. 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 (Roullet and Madec [2000], 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 EMP*S is not compensated by the advection of fluid through the top level : in the rigid lid case (contrary to the linear free surface), because *w(k=1) = 0*. As a result, even if the budget of *EMP* is zero in average over the whole ocean domain, the associated salt flux is not, as sea-surface salinity and *EMP* are intrinsically correlated (high *SSS* are found where evaporation is strong while low *SSS* is usually associated with high precipitation or river runoff input).

The Q_{ns} and *EMP* fields are defined and updated in *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*=T, 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*=F). A term is thus added to the time evolution equation of temperature (2.1d) while 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 including extinction coefficients is assumed for the downward irradiance I [Paulson and Simpson 1977] :

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

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] classification : $\xi_1 = 0.35m$, $\xi_2 = 0.23m$ 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 modified in routine *traqsr.F90*. Note that in z-coordinates, 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 central 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. At last, note that when the ocean is shallow (¡ 200 m), the part of the solar radiation can reach the ocean floor. In this case, we have chosen that all the radiation is absorbed at the last ocean level ($i.e.$ I_w is masked).

When coupling with a biology model (PISCES or LOBSTER), it is possible to calculate the light attenuation using information from the biology model. At the time of this writing, reading the light attenuation from a file is not implemented yet in the reference version.

case 4 bands and bio-coupling to add ! ! !

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

```
!-----------------------------------------------------------------------
&nambbc    !   bottom boundary condition (on temperature only)
!-----------------------------------------------------------------------
```

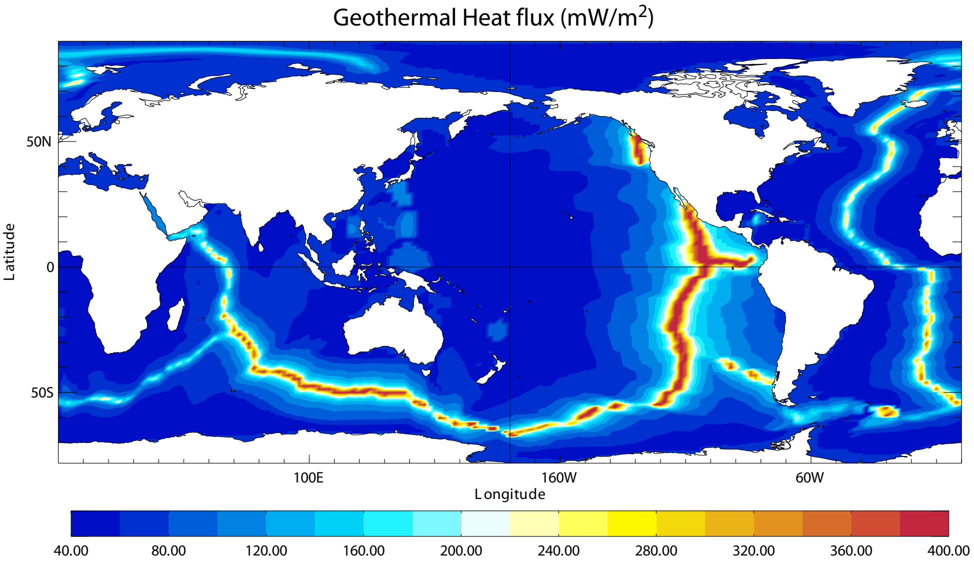

FIG. 4.2 – Geothermal Heat flux (in $mW.m^{-2}$) as inferred from the age of the sea floor and the formulae of Stein and Stein [1992].

```
!  "key_trabbc"                 ! Activate geothermal forcing (bbc)
   ngeo_flux =  2               !  = 0 no geothermal heat flux
   !                            !  = 1 constant geothermal heat flux
   !                            !  = 2 variable geothermal heat 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 considered that there is no exchange of heat nor 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. Nevertheless, there exists a non-zero heat flux across the seafloor that is associated with the solid earth cooling. This flux is weak compared with surface fluxes — a mean global value of $\sim 0.1\ W/m^2$ [Stein and Stein 1992] — but it is systematically positive and it acts only on the densest water masses. Taking this flux into account in a global ocean model increases by a few Sverdrups the deepest overturning cell (i.e. the one associated with the Antarctic Bottom Water).

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

4.5 Bottom Boundary Layer (*trabbl.F90* + key_bbl_diff or key_bbl_adv)

```
!-----------------------------------------------------------------------
&nam_trabbl   !   bottom boundary layer scheme
!-----------------------------------------------------------------------
!  "key_trabbl_dif"               !  Activate the diffusive bbl
!  "key_trabbl_adv"               !  Activate the advective bbl
   atrbbl = 10000.                !  lateral tracer coeff. for bbl scheme (m2/s)
/
```

In z-coordinate configuration, the bottom topography is represented as 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 those gravity plumes is critical to determine the density and volume flux of the densest waters of the ocean, such as the Antarctic Bottom water, or the 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 s-coordinate when the thickness of the bottom level varies in large proportions downstream of a sill [Willebrand et al. 2001], and the thickness of the plume is not resolved.

The idea of the bottom boundary layer parameterization first introduced by Beckmann and Döscher [1998] is to allow a direct communication between two adjacent bottom cells at varying level, whenever the densest water is located above the less dense water. The communication can be by diffusive fluxes (diffusive BBL), advective fluxes (advective BBL) or both. Only tracers are modified, not the velocities. Implementing a BBL parameterization for momentum is a more complex problem because of the pressure gradient errors.

4.5.1 Diffusive Bottom Boundary layer (*trabbl.F90*)

The lateral diffusivity A_l^σ in the BBL can be prescribed with a spatial dependence, e.g., in the conditional form

$$A_l^\sigma(i,j,t) = \begin{cases} \text{large} & if\ \nabla\rho \cdot \nabla H < 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.18}$$

The large value of the coefficient when the diffusive BBL is active is given by the namelist parameter *atrbbl.*

4.5.2 Advective Bottom Boundary Layer (*trabb_adv.F90*)

Implemented in NEMO v2.
Documentation to be added here

4.6 Tracer damping (*tradmp.F90*)

```
!-----------------------------------------------------------------------
&namdmp   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              !  =1 create a damping.coeff NetCDF file
   nmldmp =    1              !  type of damping in the mixed layer
   !                          !     =0 damping throughout the water column
   !                          !     =1 no damping in the mixing layer (avt >5cm2/s )
   !                          !     =2 no damping in the mixed  layer (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 in the temperature and salinity equations :

$$\begin{aligned}\frac{\partial T}{\partial t} &= \cdots - \gamma\,(T - T_o)\\ \frac{\partial S}{\partial t} &= \cdots - \gamma\,(S - S_o)\end{aligned} \qquad (4.19)$$

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 *dtacof* routine also located in *tradmp.F90*.

The two main cases in which (4.19) 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 few day time scale) whereas it is zero in the interior of the model domain. The second case corresponds to the use of the robust diagnostic method [Sarmiento and Bryan 1982]. It allows to find the velocity field consistent with the model dynamics while having a T-S field close to a given climatology field ($T_o - S_o$). The time scale associated with S_o is generally not a constant but spatially varying in order to respect some considerations. For example, it is usually set to zero in the mixed layer (defined either on a density or S_o criterion) [Madec et al. 1996] and in the equatorial region [Reverdin et al. 1991, Fujio and Imasato 1991, Marti 1992] as those two regions have a small time scale of adjustment, while smaller S_o are used in the deep ocean where the typical time scale is long [Sarmiento and Bryan 1982]. In addition it is reduced (and even 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 *nmldmp* (**namelist** *nmldmp*parameter).

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

An example of 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 coefficient and the distance to the bathymetry, respectively). Those 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       !  = 1 read the bathymetry_level
   e3zps_min =     5.      !  minimum thickness of the partial step is the min of
   e3zps_rat =     0.1     !  e3zps_min and e3zps_rat * e3t   (with 0<e3zps_rat<1)
   nmsh      =     0       !  =1 create a mesh file (coordinates, scale factors, masks)
   nacc      =     0       !  the acceleration of convergence method
   !                       !     = 0, no acceleration, rdt = rdttra
   !                       !     = 1, acceleration used, rdt < rdttra(k)
   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
   rdtmax    =  5760.      !  maximum time step on tracers
   rdth      =   800.      !  depth variation of tracer time step
   rdtbt     =    90.      !  barotropic time step (for the time splitting algorithm)
   nfice     =     5       !  frequency of ice model call
   nfbulk    =     5       !  frequency of bulk formulea call (not used if ice used)
   nclosea   =     0       !  = 0 no closed sea in the model domain
!                          !  = 1 closed sea (Caspian Sea, Great US Lakes...)
/
```

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

$$\begin{aligned} T^{t+\Delta t} &= T^{t-\Delta t} + 2\,\Delta t \ \mathrm{RHS}_T^t \\ T_f^t &= T^t \quad + \gamma \left[T_f^{t-\Delta t} - 2T^t + T^{t+\Delta t} \right] \end{aligned} \tag{4.20}$$

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 that case only the swap of arrays and the Asselin filtering is done in *tranxt.F90* module.

In order to prepare 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, 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 is the equation of state defined by the Joint Panel on Oceanographic Tables and Standards [UNESCO 1983]. It was the standard equation of state used in early releases of OPA. Even though this computation is fully vectorised, it is quite time consuming (15 to 20$\%$ of the total CPU time) as it requires the prior computation of the *in situ* temperature from the model *potential* temperature using the [Bryden 1973] polynomial for adiabatic lapse rate and a 4^th order Runge-Kutta integration scheme. Since OPA6, we have chosen the Jackett and McDougall [1995] equation of state for seawater. It allows the computation of the *in situ* ocean density directly as a function of *potential* temperature relative to the sea surface (an OPA variable), the practical salinity (another OPA 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 UNESCO [1983] and Jackett and McDougall [1995] equations of state have the same expression except that the values of the various coefficients have been adjusted by Jackett and McDougall [1995] in order to use directly 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 over ρ_o, a reference volumic mass (*rau0* defined in *phycst.F90*, usually $rau0 = 1,020\ Kg/m^3$). The default option (*neos*=0) is the Jackett and McDougall [1995] equation of state. It is highly recommended to use it. Nevertheless, for process studies, it is often convenient to use a linear approximation of the density*[1]. Two linear formulations are available : a function of T only (*neos*=1) and a function of both T and S (*neos*=2) :

$$\begin{aligned} d(T) = \rho(T)/\rho_0 \quad &= 1.028 - \alpha\, T \\ d(T,S) = \rho(T,S) \quad &= \quad \beta\, S - \alpha\, T \end{aligned} \tag{4.21}$$

[1] * 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.

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 *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.

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 of N^2 depends on the type of equation of state used (*neos* namelist parameter).

For *neos*=0 (Jackett and McDougall [1995] equation of state), the McDougall [1987] polynomial expression is used with the pressure in decibar approximated by the depth in meters :

$$N^2 = \frac{g}{e_{3w}}\ \beta(\overline{T}^{k+1/2}, \widetilde{S}, z_w) \\ \left\{ \alpha/\beta(\overline{T}^{k+1/2}, \widetilde{S}, z_w)\ \delta_{k+1/2}[T] - \delta_{k+1/2}[S] \right\} \quad (4.22)$$

where T is the *potential* temperature, $\widetilde{S} = \overline{S}^{k+1/2} - 35.$ a salinity anomaly, and α (β) the thermal (haline) expansion coefficient. Both α and β depend on *potential* temperature, salinity which are averaged at w-points prior to the computation.

When a linear equation of state is used (*neos*=1 or 2, (4.22) reduces to :

$$N^2 = \frac{g}{e_{3w}} \left(\beta\ \delta_{k+1/2}[S] - \alpha\ \delta_{k+1/2}[T] \right) \quad (4.23)$$

where α and β are the constant coefficients used to defined the linear equation of state (4.21).

4.8.3 Specific Heat (*rcp*, *phycst.F90*)

The specific heat of sea water, C_p, is a function of temperature, salinity and pressure [UNESCO 1983]. It is only used in the model to convert surface heat fluxes into surface temperature increase, thus 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 computer name is *rcp* and 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 [UNESCO 1983] :

$$\begin{aligned} T_f(S,p) = &\Big(-0.0575 + 1.710523\;10^{-3}\,\sqrt{S} - 2.154996\;10^{-4}\,S\Big)\;S \\ &- 7.53\,10^{-3}\,p \end{aligned} \tag{4.24}$$

(4.24) 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 (4.24) (last term) has 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 0 or 1 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*=T), tracers in horizontally adjacent cells generally 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). Before taking horizontal gradients between the tracers next to the bottom, a linear interpolation 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 on temperature in the i-direction, the needed interpolated temperature, $\widetilde{T}$, is :

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

and the resulting formulation of horizontal derivative and horizontal mean value of T at U-point are :

$$\begin{aligned} \delta_{i+1/2} T &= \begin{cases} \widetilde{T} \quad - T^{i} & \text{if } e_{3w}^{i+1} \geq e_{3w}^{i} \\[2ex] T^{\,i+1} - \widetilde{T} & \text{if } e_{3w}^{i+1} < e_{3w}^{i} \end{cases} \\[2ex] \overline{T}^{\,i+1/2} &= \begin{cases} (\widetilde{T} \quad - T^{\,i})/2 & \text{if } e_{3w}^{i+1} \geq e_{3w}^{i} \\[2ex] (T^{\,i+1} - \widetilde{T})/2 & \text{if } e_{3w}^{i+1} < e_{3w}^{i} \end{cases} \end{aligned} \tag{4.25}$$

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

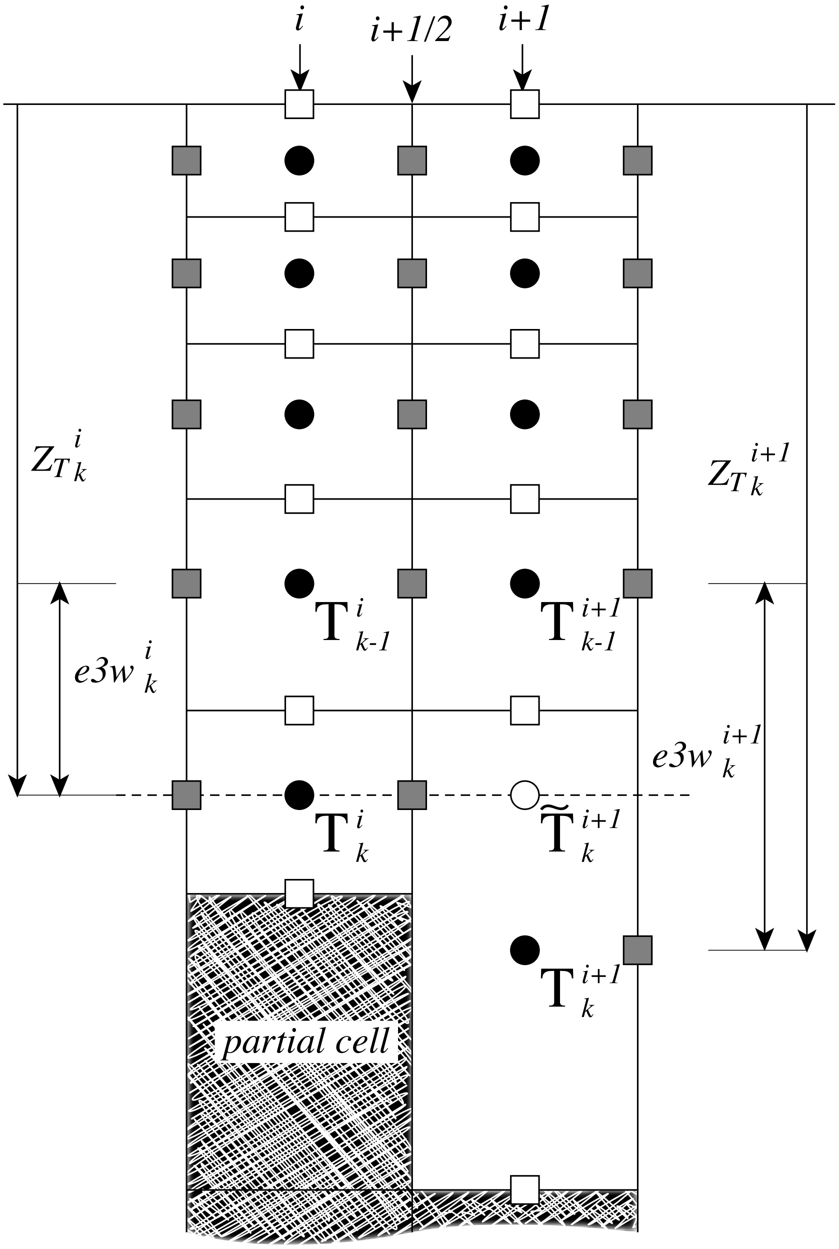

FIG. 4.3 – Discretisation of horizontal derivative and mean of tracers in z-partial step coordinate (*ln_zps*=T) in the case $(e3w_k^{i+1} - e3w_k^i) > 0$. A linear interpolation is used to estimate $\widetilde{T}_k^{i+1}$, the tracer value at the depth of the shallower tracer point of the two adjacent bottom T-points. The horizontal derivative is then given by : $\delta_{i+1/2} T_k = \widetilde{T}_k^{i+1} - T_k^i$ and the mean by : $\overline{T}_k^{i+1/2} = (\widetilde{T}_k^{i+1/2} - T_k^i)/2$.

when needed. It has to be emphasized that the computation of the interpolated density, $\widetilde{\rho}$, is not identical to the one of T and S. Instead of forming a linear approximation of density, we compute $\widetilde{\rho}$ from the interpolated value of T and S, and the pressure of at u-point (in the equation of state pressure is approximated by depth, see §4.8.1) :

$$\widetilde{\rho} = \rho(\widetilde{T}, \widetilde{S}, z_u) \quad \text{where} \;\; z_u = \min\left(z_T^{i+1}, z_T^{i}\right) \tag{4.26}$$

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

Notes : in almost all the advection schemes presented in this Chapter, both mean and derivative operators appear. Yet, it has been chosen not to use (4.25) in those schemes. : contrary to diffusion and pressure gradient computation, no correction for partial steps is applied for advection.The main motivation was to preserve the domain averaged mean variance of the field advected when using 2^{nd} order centred scheme. Sensitivity of the advection schemes to the way horizontal means are performed in the vicinity of partial cells should be further investigated in a near future.

5 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 conservative 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 mean 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 in a vorticity part (VOR), a kinetic energy part (KEG) and the vertical advection (ZAD) in the vector invariant formulation, and into Coriolis and advection (COR+ADV) in the flux formulation. The following terms are the pressure gradient contributions (HPG, Hydrostatic Pressure Gradient, and SPG, Surface Pressure Gradient). Contributions from lateral diffusion and vertical diffusion are added to the rhs in the *dynldf.F90* and *dynzdf.F90* modules ; the latter 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 of the SBC, LDF and ZDF categories and 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 accounting for 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 on the rhs of the 3D momentum equation (**key_trddyn** defined) and of the 2D barotropic vorticity balance (**key_trdvor** defined), as described in Chap. 10.

5.1 Coriolis and Advection : vector invariant form

```
!-----------------------------------------------------------------------
&nam_dynadv  !   option of physics/algorithm (not control by CPP keys)
!-----------------------------------------------------------------------
   ln_dynadv_vec   = .TRUE.    !  vector form (T) 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 most often used in applications of the ocean model. The flux form option (next section) has been introduced recently in version 2 of NEMO. 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.

5.1.1 Vorticity term (*dynvor.F90*)

```
!-----------------------------------------------------------------------
&nam_dynvor   !   option of physics/algorithm
!-----------------------------------------------------------------------
   ln_dynvor_ene = .false.    !  enstrophy conserving scheme
   ln_dynvor_ens = .true.     !  energy conserving scheme
   ln_dynvor_mix = .false.    !  mixed energy/enstrophy conserving scheme
   ln_dynvor_een = .false.    !  energy and enstrophy scheme
/
```

Different discretisations of the vorticity term (selected by the *ln_dynvor_xxx* namelist variable to true) are available, that conserve potential enstrophy of horizontally non-divergent flow, horizontal kinetic energy, or 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 (s-coordinate or partial step topography), but note that in full step z-coordinate (**key_zco** defined), $e_{3u} = e_{3v} = e_{3f}$ so that the vertical scale factors disappear.

enstrophy conserving scheme (*ln_dynvor_ens*=T)

In this case, 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 of the total kinetic energy. It is given by :

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

energy conserving scheme (*ln_dynvor_ene*=T)

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

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

mixed energy/enstrophy conserving scheme (*ln_dynvor_mix*=T)

In this case, a mixture of the two previous schemes is used. It consists of the enstrophy conserving scheme (5.1) applied to the relative vorticity term and of the horizontal kinetic energy conserving scheme (5.2) applied to the planetary vorticity term.

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

energy and enstrophy conserving scheme (*ln_dynvor_een*=T)

In this case, the vorticity term is evaluated using the vorticity advection scheme of Arakawa and Hsu [1990]. This scheme conserves both total energy and potential enstrophy in the limit of horizontally nondivergent flow (i.e. χ=0), While this scheme is more complicated that the vorticity advection scheme of Sadourny [1975] and does not conserve potential enstrophy and total energy in general flow, as does a scheme from Arakawa and Lamb [1981], it tolerates arbitrarily thin layers. This feature is essential for simulating either outcropping isopycnals or large amplitude topography.

The Arakawa and Hsu [1990] vorticity advection scheme for a single layer is modified for spherical coordinates as described by Arakawa and Lamb [1981].

The potential vorticity, defined at 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

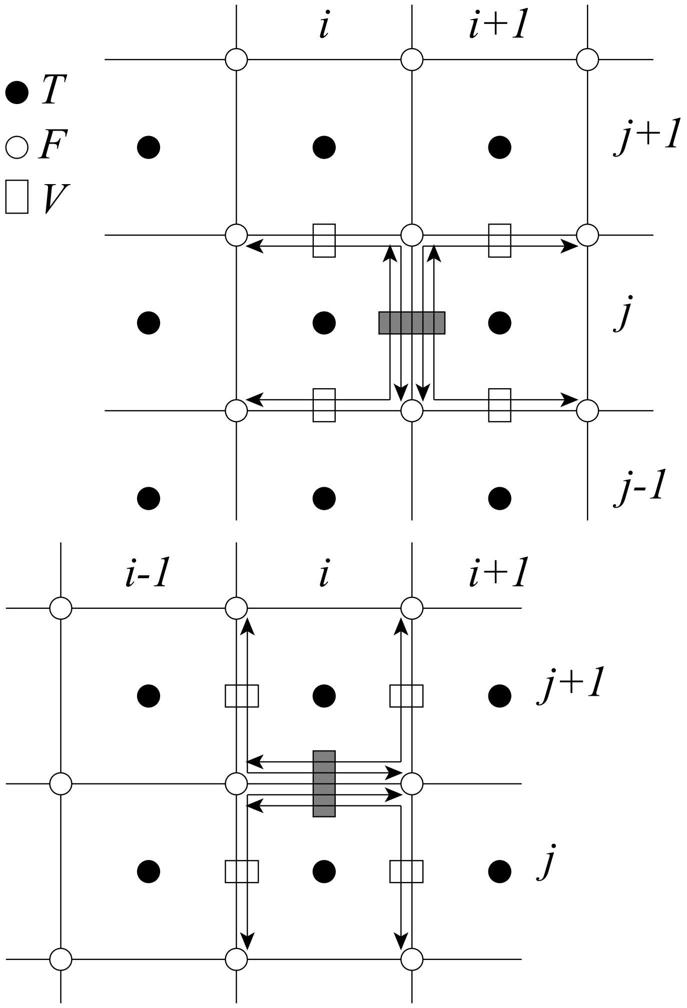

FIG. 5.1 – Triads used in the energy and enstrophy conserving scheme (een) for u-component (upper panel) and v-component (lower panel).

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.

The vorticity terms are represented as :

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

where a, b, c and d are 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*)

There is a single discrete formulation of the kinetic energy gradient term, that conserves the total kinetic energy together with the formulation chosen for the vertical advection (see below).

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

5.1.3 Vertical advection term (*dynzad.F90*)

The discrete formulation of the vertical advection term conserves the total kinetic energy together with the formulation chosen for the gradient of kinetic energy (KE). 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 Annexe 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  !   option of physics/algorithm (not control by CPP keys)
!-----------------------------------------------------------------------
   ln_dynadv_vec    = .TRUE.     !  vector form (T) 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 :

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

Two advection schemes (2^{nd} order centered finite difference scheme, CEN2, or a 3^{rd} order upstream biased scheme, UBS). The latter is described in Shchepetkin and McWilliams [2005]. The schemes are selected using the namelist variable *ln_dynadv_xxx* In flux form, the advection 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 and f-, T- and vw-points for u and v, respectively.

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

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*=T)

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 [Shchepetkin and McWilliams 2005]. The overall performance of the advection scheme is similar to that reported in Farrow and Stevens [1995]. It is a relatively good compromise between accuracy and smoothness. It is not a *positive* scheme, meaning that false extrema are permitted but their amplitude is significantly reduced over the centred second order method.

The UBS scheme is not used in all the direction. In the vertical, it has been preferred to keep the centred 2^{nd} order evaluation of the advection, i.e. u_{uw}^{ubs} and u_{vw}^{ubs} in (5.12) are used. Indeed, UBS is diffusive, it is associated with vertical mixing of momentum. Since vertical mixing of momentum is source term of the TKE equation…

For stability reasons, in (5.13), the first term 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 Webb et al. [1998] in the context of the Quick advection scheme.

NB 1 : 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 [Webb et al. 1998]. 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.

NB 2 : In the current version of *dynadv_ubs.F90*, there is also a 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:
   !                          !     = T : semi-implicit time scheme
   !                          !     = f : centered      time scheme
   nn_dynhpg_rst   =  0       !  add dynhpg implicit variables in restart ot not (1/0)
/
```

Suppress the namflg namelist and incorporate it in the namhpg namelist

The key distinction between the different algorithms 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* velocity), 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*=T)

The hydrostatic pressure can be obtained by integrating the hydrostatic equation vertically from the surface. Nevertheless, the pressure is quite large at great depth while its horizontal gradient is several orders of magnitude smaller. This may lead to large truncation error 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*=T)

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

It is necessary to do so at the last ocean level ; otherwise the horizontal hydrostatic pressure gradient evaluated in z-coordinate with partial step is exactly as in 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 vertically temperature and salinity before computing the density. Horizontal gradients of 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 *s-z* coordinates

Pressure gradient formulations in s coordinates have been the subject of a vast literature ($e.g.$, Song [1998], Shchepetkin and McWilliams [2003]). A number of different pressure gradient options are coded, but they are not yet fully documented nor tested.

Traditional coding (see for example Madec et al. [1996] : (*ln_dynhpg_sco*, *ln_dynhpg_hel*)

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

Where the first term is the pressure gradient along coordinates, computed as in (II.3.2), and z_T is the depth of T-point evaluated from the sum of the vertical scale factor at W-point (e_{3w}). The version *ln_dynhpg_hel* 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) [Song 1998] (*ln_dynhpg_wdj*=T)

Density Jacobian with cubic polynomial scheme (djc) [Shchepetkin and McWilliams 2003] (*ln_dynhpg_djc*=T)

Rotated axes scheme (rot) [Thiem and Berntsen 2006] (*ln_dynhpg_rot*=T)

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 surface are not horizontal but follow the free surface [Levier et al. 2007]. The other pressure gradient options are not yet available.

5.3.4 Time-scheme (*ln_dynhpg_imp*=T/F)

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 a ocean domain including weakly stratified regions) the physical phenomena that control the time-step are internal gravity waves (IGWs). A semi-implicit scheme for doubling the stability limit associated with IGWs can be used [Brown and Campana 1978, Maltrud et al. 1998]. It consists in evaluating 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 standard leapfrog scheme.

leapfrog scheme (*ln_dynhpg_imp*=F) :

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

semi-implicit scheme (*ln_dynhpg_imp*=T) :

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

The semi-implicit time scheme (5.18) is made possible without significant additional computation as 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 Courant-Friedrichs-Lewy (CFL) limit associated to the hydrostatic pressure gradient double using (5.18) compared to 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 CFL limits associated with advection or diffusion.

In practice, the semi-implicit scheme is used when *ln_dynhpg_imp*=T. 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 :

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

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

5.4 Surface pressure gradient (*dynspg.F90*)

```
!-----------------------------------------------------------------------
&nam_dynspg    !   surface pressure gradient   (CPP key only)
!-----------------------------------------------------------------------
!  "key_vvl"                    !  Activate the variable volume level
!  "key_dynspg_exp"             !  explicit free surface
!  "key_dynspg_flt"             !  filtered free surface
!  "key_dynspg_ts"              !  split-explicit free surface
!  "key_dynspg_rl"              !  rigid-lid
/
```

The surface pressure gradient term is related to 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 active). In the linear free surface case (§2.2.2) and rigid lid (§2.2.3), the vertical scale factors e_3 are fixed in time, while 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, which is a modification of the continuous equations (see (2.6)), and the split-explicit free surface 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 small enough to describe the external gravity waves (typically a few ten 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 (evaporation minus precipitation, and minus river runoffs (if the later are introduced as a surface freshwater flux, see §6) expressed in $Kg.m^{-2}.s^{-1}$, and $\rho_w = 1,000\,Kg.m^{-3}$ is the volumic mass of pure water. 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 $\left.\rho\right|_{k=1} / \rho_o$ factor is omitted in (5.21).

Split-explicit time-stepping (key_dynspg_ts)

```
!-----------------------------------------------------------------------
&namdom   !   space and time domain (bathymetry, mesh, timestep)
!-----------------------------------------------------------------------
   ntopo     =     1       !  = 1 read the bathymetry_level
   e3zps_min =     5.      !  minimum thickness of the partial step is the min of
   e3zps_rat =     0.1     !  e3zps_min and e3zps_rat * e3t  (with 0<e3zps_rat<1)
   nmsh      =     0       !  =1 create a mesh file (coordinates, scale factors, masks)
   nacc      =     0       !  the acceleration of convergence method
   !                       !     = 0, no acceleration, rdt = rdttra
   !                       !     = 1, acceleration used, rdt < rdttra(k)
   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
   rdtmax    =  5760.      !  maximum time step on tracers
   rdth      =   800.      !  depth variation of tracer time step
   rdtbt     =    90.      !  barotropic time step (for the time splitting algorithm)
   nfice     =     5       !  frequency of ice model call
   nfbulk    =     5       !  frequency of bulk formulea call (not used if ice used)
   nclosea   =     0       !  = 0 no closed sea in the model domain
!                          !  = 1 closed sea (Caspian Sea, Great US Lakes...)
/
```

The split-explicit free surface formulation used in OPA follows the one proposed by Griffies [2004]. The general idea is to solve the free surface equation with a small time step, while the three dimensional prognostic variables are solved with a longer time step that is a multiple of *rdtbt* (Figure III.3).

The split-explicit formulation has a damping effect on external gravity waves, which is weaker than the filtered free surface but still significant as shown by Levier et al. [2007] in the case of an analytical barotropic Kelvin wave.

Filtered formulation (key_dynspg_flt)

The filtered formulation follows the Roullet and Madec [2000] 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

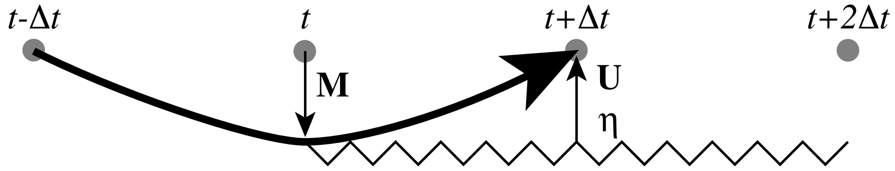

FIG. 5.2 – Schematic of the split-explicit time stepping scheme for the barotropic and baroclinic modes, after Griffies [2004]. Time increases to the right. Baroclinic time steps are denoted by $t - \Delta t$, $t, t + \Delta t$, and $t + 2\Delta t$. The curved line represents a leap-frog time step, and the smaller barotropic time steps $N\Delta t = 2\Delta t$ are denoted by the zig-zag line. The vertically integrated forcing **M**(t) computed at baroclinic time step t represents the interaction between the barotropic and baroclinic motions. While keeping the total depth, tracer, and freshwater forcing fields fixed, a leap-frog integration carries the surface height and vertically integrated velocity from t to $t + 2\Delta t$ using N barotropic time steps of length Δt. Time averaging the barotropic fields over the N+1 time steps (endpoints included) centers the vertically integrated velocity at the baroclinic timestep $t + \Delta t$. A baroclinic leap-frog time step carries the surface height to $t + \Delta t$ using the convergence of the time averaged vertically integrated velocity taken from baroclinic time step t.

the namelist variable *rnu*. The default value is 1, as recommended by Roullet and Madec [2000]

rnu=1 to be suppressed from namelist !

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 [Levier et al. 2007] 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 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 :

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

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  (+ "key_ldfslp" if ln_sco=T)
   ln_dynldf_iso    =  .false.   !     iso-neutral (+ "key_ldfslp")
   !                           !  Coefficient
   !  "key_ldfdyn_c1d"         !     Ahm = F(k)
   !  "key_ldfdyn_c2d"         !     Ahm = F(i,j)
   !  "key_ldfdyn_c3d"         !     Ahm = F(i,j,k)
   ahm0     = 40000.           !     horizontal eddy viscosity for the dynamics (m2/s)
   ahmb0    =     0.           !     background eddy viscosity for isopycnal diffusion (m2/s)
```

The options available for lateral diffusion are 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 lateral physics (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 following user's choice (see Chap. §7).

5.5.1 Iso-level laplacian operator (*ln_dynldf_lap*=T)

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

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

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 pure z-coordinate (**key_zco** defined), $e_{3u} = e_{3v} = e_{3f}$ so that they disappear from the rotational part of (5.24).

5.5.2 Rotated laplacian operator (*ln_dynldf_iso*=T)

A rotation of lateral momentum diffusive operator is needed for isoneutral diffusion in z-coordinates (*ln_dynldf_iso*=T) and for either isoneutral (*ln_dynldf_iso*=T) or geopotential (*ln_dynldf_hor*=T) diffusion in s-coordinates. In the partial step case, coordinates are horizontal excepted at the deepest level and no rotation is performed when *ln_dynldf_hor*=T. The diffusive operator is defined simply as the divergence of down gradient momentum fluxes on each momentum components. 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 lateral physics chapter (Chap. §8).

5.5.3 Iso-level bilaplacian operator

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).

add a remark on the the change in the position of the coefficient

5.6 Vertical diffusion term (*dynzdf.F90*)

```
!-----------------------------------------------------------------------
&namzdf   !   vertical physics
!-----------------------------------------------------------------------
   ln_zdfnpc = .false.        !  Non-Penetrative Convection
   avm0      = 1.2e-4         !  Kz on momemtum (m2/s)
   !                          !  (background Kz if not "key_zdfcst")
   avt0      = 1.2e-5         !  Kz for tracers (m2/s)
   !                          !  (background Kz if not "key_zdfcst")
   ln_zdfevd = .true.         !  enhanced vertical diffusion
   avevd     =   100.         !  Kz for enhanced diffusion scheme (m2/s)
   n_evdm    =     0          !  enhanced mixing Kz apply on tracer (=0)
   !                          !      or on both tracer and momentum (=1)
   ln_zdfexp =  .false.       !  =T/F  split explicit / implicit
   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 a too restrictive constraint on the time step in a pure explicit time stepping case. Two time stepping schemes can be used for the vertical diffusion term : (a) a forward time differencing scheme (*ln_zdfexp*=T) using a time splitting technique (*n_zdfexp* > 1) or (b) a backward (or implicit) time differencing scheme (*ln_zdfexp*=F) (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 diffusive operators given by (2.42) take the following semi-discrete space form :

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

where A_{uw}^{vm} andA_{vw}^{vm} are the vertical eddy viscosity and diffusivity coefficients. The way these coefficients can be 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,

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

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 is specified through a bottom friction parameterization (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       !  = 1 read the bathymetry_level
   e3zps_min =      5.      !  minimum thickness of the partial step is the min of
   e3zps_rat =      0.1     !  e3zps_min and e3zps_rat * e3t  (with 0<e3zps_rat<1)
   nmsh       =     0       !  =1 create a mesh file (coordinates, scale factors, masks)
   nacc       =     0       !  the acceleration of convergence method
   !                        !     = 0, no acceleration, rdt = rdttra
   !                        !     = 1, acceleration used, rdt < rdttra(k)
   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
   rdtmax     =  5760.      !  maximum time step on tracers
   rdth       =   800.      !  depth variation of tracer time step
   rdtbt      =    90.      !  barotropic time step (for the time splitting algorithm)
   nfice      =     5       !  frequency of ice model call
   nfbulk     =     5       !  frequency of bulk formulea call  (not used if ice used)
   nclosea    =     0       !  = 0 no closed sea in the model domain
!                           !  = 1 closed sea (Caspian Sea, Great US Lakes...)
/
```

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

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

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 nly the swap of arrays and Asselin filter 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 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 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 z-coordinate with full step (**key_zco** defined), $e_{3u} = e_{3v} = e_{3f}$ so that they disappear from (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 respectively for the nonlinear advection and the computation of the vertical velocity.

5.9.2 Vertical velocity (*wzvmod.F90*)

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

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

With a free surface, the top vertical velocity is non-zero, due the freshwater forcing and the variations of the free surface elevation. When the free surface is linear 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** defined), the surface boundary condition ($\left. w \right|_{\text{surface}} = 0$) is automatically achieved at least at the computer accuracy, due to the discrete expression of the surface pressure gradient (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 Levier et al. [2007], report available on the NEMO web site).

6 Surface Boundary Condition (SBC)

Contents

At the time of this writing, the new surface module that is described in this chapter (SBC) is not yet part of the current distribution. The current way to specify the surface boundary condition is such a mess that we did not attempt to describe it. Nevertheless, apart from the way the surface forcing is implemented, the information given here are relevant for a NEMO v2.3 user.

The ocean needs 7 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 $(\text{EMP}\ ,\ \text{EMP}_S)$

The river runoffs (RUNOFF)

Four different ways are offered to provide those 7 fields to the ocean : an analytical formulation, a flux formulation, a bulk formulae formulation (CORE or CLIO bulk formulae) and a coupled formulation (exchanges with a atmospheric model via OASIS coupler). In addition, the resulting fields can be further modified on used demand via several namelist option. These option control the addition of a surface restoring term to observed SST and/or SSS, the modification of fluxes below ice-covered area (using observed ice-cover or a sea-ice model), the addition of river runoffs as surface freshwater fluxes, and the addition of a freshwater flux adjustment on order to avoid a mean sea-level drift.

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. Finally, the different options that modify the fluxes inside 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. Their two components are assumed to be interpolated on the ocean mesh, i.e. provided at U- and V-points and projected onto the (**i**,**j**) referential. They are applied as a surface boundary condition of the computation of the momentum vertical mixing trend (**dynzdf** 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 in two parts, a non solar and solar heat fluxes. The former is the non penetrative part of the heat flux (i.e. sensible plus latent plus long wave heat fluxes). It is applied as a surface boundary condition trend of the first level temperature time evolution equation (*trasbc.F90* module).

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

The latter 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*=T.

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

where I_w is an adimensional function that describes the way the light penetrates inside the water column. It is generally a sum of decreasing exponential (see §4.4.2).

The surface freshwater budget is provided through two non-necessary identical fields EMP and EMP_S. Indeed, a surface freshwater flux has two effects : it changes the volume of the ocean and it changes the surface concentration of salt (an others tracers). Therefore it appears in the sea surface height and salinity time evolution equations as a volume flux, EMP (*dynspg_xxx* modules), and concentration/dilution effect, EMP$_S$ (*trasbc.F90* module), respectively.

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

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

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

When a sea-ice model is considered, the water exchanged between ice and ocean is not fresh as mean ice salinity is ∼*4 psu*. In this case, EMP$_S$ take into account both concentration/dilution effect associated with freezing/melting together with 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 (not impact on EMP) while it modifies the SSS (see § on LIM sea-ice model).

Note that SST can also be modified by a freshwater flux. Precipitations (in particular solid one) may have a temperature significantly different from the SST. Due to the lack of information about the temperature of precipitations, 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 not heat flux associated with precipitation ! An excess of precipitation will change the ocean heat content and is therefore associated with a heat flux (not diagnosed in the model) [Roullet and Madec 2000]).

Miss :

A extensive description of all namsbc namelist (parameter that have to be created !)

Especially the *nf_sbc*, the *sbc_oce.F90* module (fluxes + mean sst sss ssu ssv) i.e. information required by flux computation or sea-ice

Add nqsr = 0 / 1 replace key_traqsr

sbc_oce.F90 containt the definition in memory of the 7 fields (6+runoff), add a word on runoff : included in surface bc or add as lateral obc. . . .

Sbcmod manage the "providing" (fourniture) to the ocean the 7 fields

Fluxes update only each nf_sbc time step (namsbc) explain relation between nf_sbc and nf_ice, do we define nf_blk ? ? ? ? only one nf_sbc

Explain here all the namlist namsbc variable. . . .

End Miss

The ocean model provides the following variables averaged over nf_sbc time-step :

The mean computation is done in sbcmod (

Penser a mettre dans le restant l'info nf_sbc ET nf_sbc*rdt de sorte de reinitialiser la moyenne si on change la frequence ou le pdt

NB : creer cn_sbc_ice (cn_ = character in the namelist) with 3 cases :

= 'noice' no specific call

Variable desciption	Computer name	Units	point
i-component of the surface current	ssu_u	$m.s^{-1}$	U
j-component of the surface current	ssv_m	$m.s^{-1}$	V
Sea surface temperature	sst_m	$^\circ K$	T
Sea surface salinty	sss_m	psu	T

= 'iceif ' "ice-if" sea ice, i.e. read observed ice-cover and modified sbc bellow those area.

= 'lim' LIM sea-ice model is called which update the sbc fields in ice covered area

? modify the nsbc_ice variable depending of this parameter (from –1, 0 to 1) End Penser a

6.2 Analytical formulation (*sbcana* module)

```
!-----------------------------------------------------------------------
&namtau    !   surface wind stress
!-----------------------------------------------------------------------
   ntau000 =      0        !  gently increase the stress over the first ntau_rst time-steps
   tau0x   =      0.e0     !  uniform value used as default surface heat flux
   tau0y   =      0.e0     !  uniform value used as default solar radiation flux
/

!-----------------------------------------------------------------------
&namflx    !   surface fluxes
!-----------------------------------------------------------------------
   q0      =      0.e0     !  uniform value used as default surface heat flux
   qsr0    =      0.e0     !  uniform value used as default solar radiation flux
   emp0    =      0.e0     !  uniform value used as default surface freswater budget (E-P)
   dqdt0   =      -40.     !  feedback coefficient for SST damping (W/m2/K)
   deds0   =        0.     !  feedback coefficient for SSS damping (mm/day)
/
```

The analytical formulation of the surface boundary condition is set by default. In this case, all the 6 fluxes needed by the ocean are assumed to be uniform in space. They take constant values given in the namlist namsbc_ana : *utau0*, *vtau0*, *qns0*, *qsr0*, *emp0* and *emps0*. while the runoff is set to zero. In addition, the wind is allowed to reach its nominal value within a given number of time step (*ntau000*).

If a user wants to applied a different analytical forcing, *sbcana.F90* module is the very place to do that. As an example, one can have a look to the *sbc_ana_gyre.F90* routine which provides the analytical forcing of the GYRE configuration (see GYRE configuration manual, in preparation).

6.3 Flux formulation (*sbcflx.F90* module, key_sbcflx)

In the flux formulation (**key_sbcflx** defined), 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, and a logical setting whether a time interpolation to the model time step is asked are not for this field). (fld_i namelist structure).

Describe the information given ?

Add an info about on-line interpolation or not ? at with which scale...

Caution : when the frequency is set to –12, the data are monthly values. There are assumed to be climatological values, so time interpolation between December the 15^{th} and January the 15^{th} is done using record 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. These file must only content a single record. If they don't exist, the will assume that the previous year last record is equal to the first record of the previous 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 damping term to observed SST and/or SSS. See §6.6.1 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 depends on the bulk formulae used. Two of them are available : the CORE and CLIO bulk formulea. The choice is made by activating the CPP key **key_sbcblk_core** or **key_sbcblk_clio**, respectively.

Note : if a sea-ice model is used then blah blah blah...

CORE bulk formulea

The CORE bulk formulae have been developed by Large and Yeager [2004]. They have been design 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 meter wind speed, air temperature and specific humidity).

The required 8 input fields are :

Note that the air velocity can be provided at either tracer ocean point or velocity ocean point.

Explain low resolution, better to provide it at U-V, high resolution better at T-point... Explain why, scheme ?

Add a namelist parameter to provide a switch from U/V or T (or I ? ?) point for utau/vtau

CLIO bulk formulea

Variable desciption	Computer name	Units	point
i-component of the 10m air velocity	utau	$m.s^{-1}$	T or U
j-component of the 10m air velocity	vtau	$m.s^{-1}$	T or V
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

The CLIO bulk formulae have been developed several years ago for the Louvain-la-neuve coupled ice-ocean model (CLIO, Goosse et al. 1997). It is a simpler bulk formulae that assumed the stress to be known and computes the radiative fluxes from a climatological cloud cover.

The required 7 input fields are :

Variable desciption	Computer name	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
Secific 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, the input data information required by the model is provided in the namsbc_blk_core or namsbc_blk_clio namelist (via the structure fld_i). The same assumption is made about the value of the first and last record in each file.

6.5 Coupled formulation (*sbccpl.F90* module)

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.

6.6 Miscellanea options

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

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

$$Q_{ns} = Q_{ns}^o + \frac{dQ}{dT} \left(\left. T \right|_{k=1} - SST_{Obs} \right) \tag{6.5}$$

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 $-40\ W.m^{-2}.\,^\circ\mathrm{K}^{-1}$. 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 fits SST then Q is equal to Q_o.

In the fresh water budget, a feedback term can also be added :

$$EMP = EMP_o + \gamma_s^{-1} \left(S - SSS_{Obs} \right) |S \tag{6.6}$$

where EMP$_o$ is a net surface fresh water flux (observed, climatological or atmospheric model product), *SSS*$_{Obs}$is a sea surface salinity (usually a time interpolation of the monthly mean PHC climatology [Steele et al. 2001], S 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 (III.4.4) as the atmosphere does not care about ocean surface salinity [Madec and Delecluse 1997]. The SSS restoring term can only be view as a flux correction on freshwater fluxes to reduce the uncertainties we have on the observed freshwater budget.

6.6.2 Handling of ice-covered area

The presence of sea-ice at the top of the ocean strongly modify the surface fluxes

The presence at the sea surface of an ice cover area modified all the fluxes transmitted to the ocean. There is two cases whereas a sea-ice model is used or not.

Without sea ice model, the information of ice-cover / open ocean is read in a file (either the directly the ice-cover or the observed SST from which ice-cover is deduced using a criteria on freezing point temperature).

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

It is convenient to introduce the river runoff in the model as a surface fresh water fluxes. … blah blah. …

Nevertheless, Pb of vertical resolution and increase of Kz in vicinity of river mouths…

Control of the mean sea level

6.6.4 Freshwater budget control (*sbcfwb.F90*)

```
!------------------------------------------------------------------------
&namfwb   !   freshwater budget correction
!------------------------------------------------------------------------
   ln_fwb    = .true.      !  flag for freshwater budget correction (0 annual mean)
/
```

freshwater budget correction. . .

7 Lateral Boundary Condition (LBC)

Contents

7.1 Boundary Condition at the Coast (*shlat*)

```
!-----------------------------------------------------------------------
&namlbc    !   lateral momentum boundary condition
!-----------------------------------------------------------------------
! "key_noslip_accurate"     ! Activate accurate no-slip boundary condition
   shlat   =      2.        !      shlat = 0 :           free slip
                            !  0 < shlat < 2 : "partial" free-slip
                            !      shlat = 2 :           no slip
                            !  2 < shlat     : "strong"  no-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 do loop. An efficient way to work over

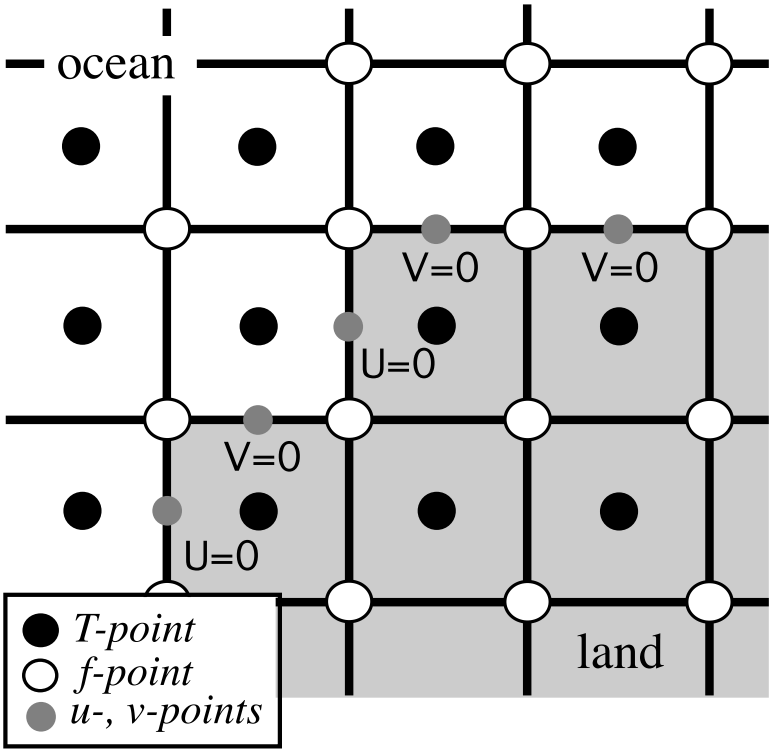

FIG. 7.1 – Lateral boundary (thick line) at T-level. The velocity normal to the boundary is set to zero.

the whole domain while specifying the boundary conditions is to use the multiplication by mask arrays in the computation. A mask array is a matrix which elements are 1in 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 settled by multiplying variables by the right 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 u-point, ensures that the heat flux is zero inside land and at the boundaries as mask$_u$ is zero at solid boundaries defined at u-points in this case (normal velocity u remains zero at the coast) (Fig. 7.1).

On momentum the situation is a bit more complex as two boundary conditions must be provided along the coast (one on the normal velocity and the other on the tangential velocity). The boundary of the ocean in C-grid is defined by the velocity-faces. For example, at a given T-level, the lateral boundary (coastline or 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 settled by the mask system. The boundary condition on the tangential velocity requires a more specific treatment. It influences the relative vorticity and momentum diffusive trends, and is only required to compute the vorticity at the coast. Four different type of the lateral boundary condition are available, controlled by the value of *shlat* namelist parameter (which is equal to the value of the mask$_f$ array along the coastline) :

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 closest land velocity gridpoint were of the same magnitude as the closest ocean velocity gridpoint 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 allows to provide a vorticity field computed with the no-slip boundary condition simply by 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 vanish the tangential velocity at the coast (Fig. 7.1-c). This can be settled 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-coordinates (pure s-coordinate), the lateral boundary condition on momentum tangential velocity is of much little 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 Shchepetkin and O'Brien [1996] is also available through the activation of **key_noslip_accurate**. It is based on a fourth order evaluation of the shear at the

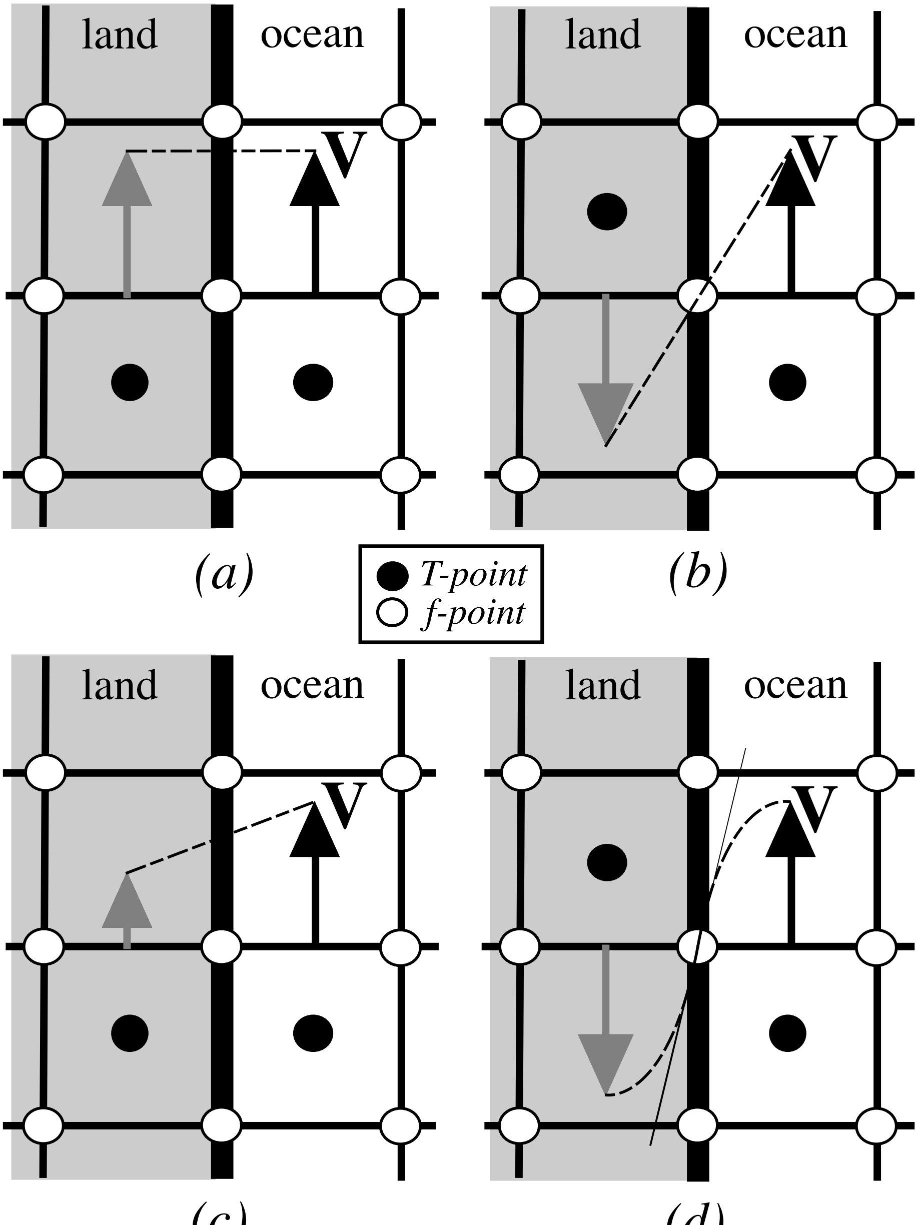

FIG. 7.2 – lateral boundary condition (a) free-slip ($shlat = 0$) ; (b) no-slip ($shlat = 2$) ; (c) "partial" free-slip ($0 < shlat < 2$) and (d) "strong" no-slip ($2 < shlat$). Implied "ghost" velocity inside land area is display in grey.

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). Shchepetkin and O'Brien [1996] found that such a technique considerably improves the quality of the numerical solution. In *NEMO*, the improvement have not been found so spectacular in the half-degree global ocean (ORCA05), but significant reduction of numerically induced coast upwellings were found in eddy resolving simulation of the Alboran Sea [Olivier 2001]. Nevertheless, as no-slip boundary condition is not recommended in eddy permitting or resolving simulation [**?**], the use of this option is 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 to qualify the nature of coastline grid point (convex or concave corners, straight north-south or east-west coast) which is performed in *domask.F90* module, *dom_msk_nsa* routine.

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 *par_oce.F90* file. Each time such a boundary condition is needed, it is set by a call to *lbclnk.F90* routine. The computation of momentum and tracer trends proceed from $i = 2$ to $i = jpi - 1$ and from $j = 2$ to $j = jpj - 1$, $i.e.$in the inner model domain. 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) while first column is set to the value of the before last column and 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 neither option for north-south cyclic nor 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-points 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 on massively parallel computer (**key_mpp** defined).

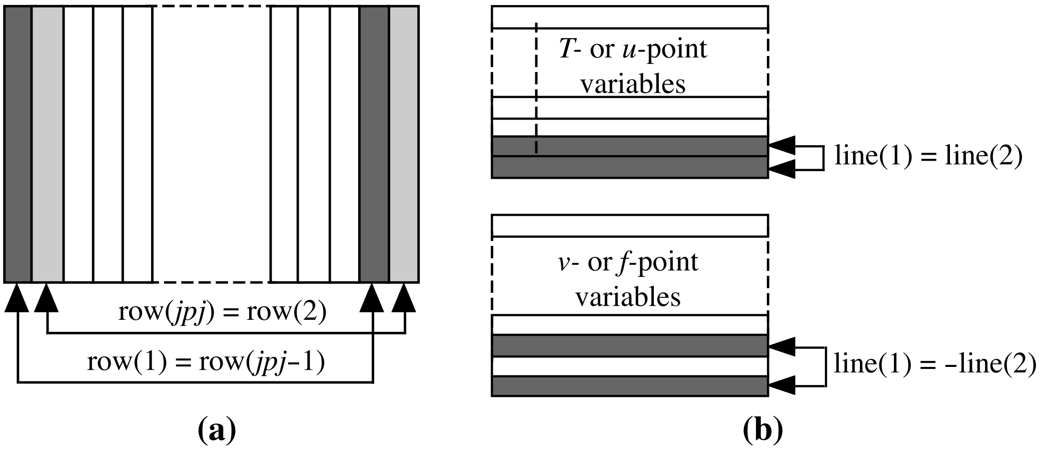

FIG. 7.3 – setting of (a) east-west cyclic (b) symmetric across the equator boundary conditions.

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

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

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

For massively parallel processing (mpp), a domain decomposition method is used. The basis of the method consists in splitting the large computation domain of a numerical experiment into several smaller domains and solving 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 explicitly organized by specific statements (message passing method).

A big advantage is that the method does not need many modifications of the initial FORTRAN code. For 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 presents more scalability when the number of processors is increased. The porting of OPA code on a 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 performances on a T3D and T3E Cray computers have been made in collaboration with IDRIS and

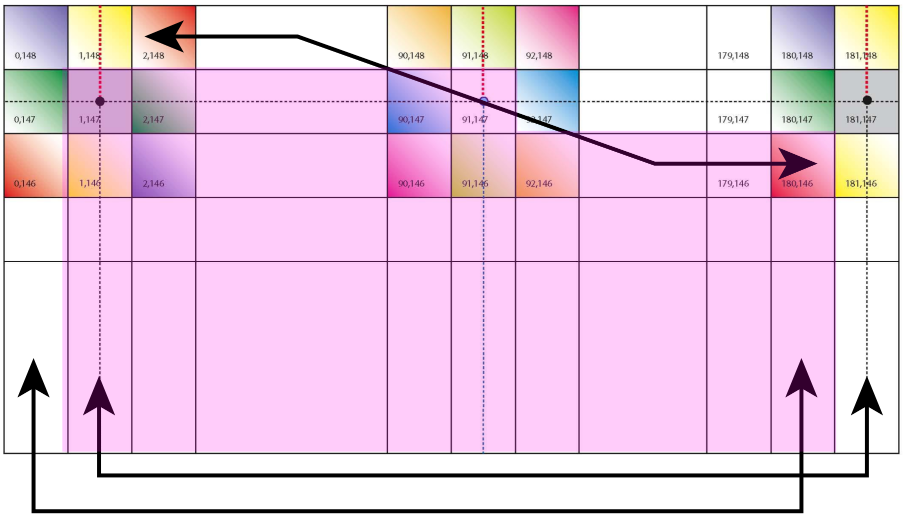

FIG. 7.4 – North fold boundary with a T-point pivot and cyclic east-west boundary condition ($jperio = 4$), as used in ORCA 2, 1/4, and 1/12. Pink shaded area corresponds to the inner domain mask (see text).

CNRS. The present implementation is largely inspired from Guyon's work [Guyon 1995].

The parallelization strategy is defined by the physical characteristics of the ocean model. Second order finite difference schemes leads to local discrete operators that depend at the very most on one neighbouring point. The only non-local computations concerne 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 stocks lateral boundary conditions for computations in the inner sub-domain. The overlapping area is reduced to one row. After a computation, a communication phase starts : each processor sends to its neighbouring processors the update values of the point corresponding to the overlapping area of its neighbouring sub-domain. 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 cumputers. More specific languages (i.e. computer dependant languages) can be easily used to speed up the communication, such as SHEM on 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 MPP computer (**key_mpp_mpi** defined). It has to be noticed that when using

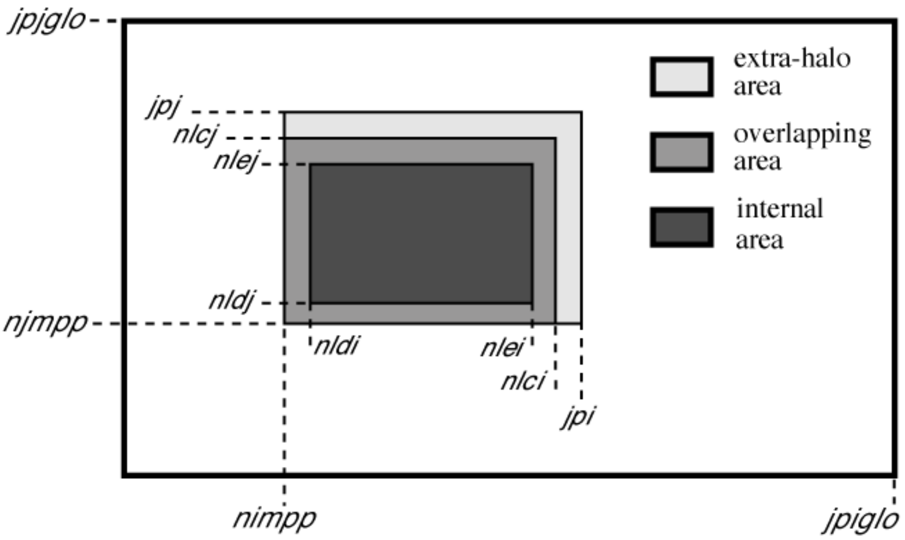

FIG. 7.5 – Positioning of a sub-domain when massively parallel processing is used.

MPP version of the model, the east-west cyclic boundary condition is implicitly done, while 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 at its own local memory. For this reason, the main model dimensions are now the local dimensions of the subdomain (pencil) that are noted *jpi*, *jpj*, *jpk*. These dimensions include the internal domain and the overlapping rows. The number of overlapping rows 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 :

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

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 defines also 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 element of T_g, a global array (whole domain) by the relationship :

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

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 situation 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 OPA model computes equation terms with the help of mask arrays (0 onto land points and 1 onto sea points). It is easily readable and very efficient in the context of the vectorial architecture. But in the case of scalar processor, computations over the land regions becomes more expensive in term of CPU time. It is all the more 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 allows to search in the mpp domain decomposition strategy if a splitting can be found with a maximum number of only land points processors which could be eliminated (mppini2 program). This optimisation is made with the knowledge of the specific bathymetry in a first time and after, the OPA model, in its initialization part, take account only processors with a sea region. For that, one must indicate in the parameter file the initial cutting along i- and j-axes with jpni and jpnjand the ocean processor number jpnij ¡ jpni x jpnj. Each processor name and neighbour parameters (nbound, nono, noea,...) are modified by an algorithm in the inimpp2.F subroutine.

The OPA model computes equation terms with the help of mask arrays (0 onto land points and 1 onto sea points). It is easily readable and very efficient in the context of the vectorial architecture. But in the case of scalar processor, computations over the land regions becomes more expensive in term of CPU time. It is all the more so 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 allows to search in the mpp domain decomposition strategy if a splitting can be found with a maximum number of only land points processors which could be eliminated : the mpp_optimiz tools (available from the DRAKKAR web site). This optimisation is made with the knowledge of the specific bathymetry. The user 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 set each processor name and neighbour parameters (nbound, nono, noea,...) so that the land processors are not taken into account.

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

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).

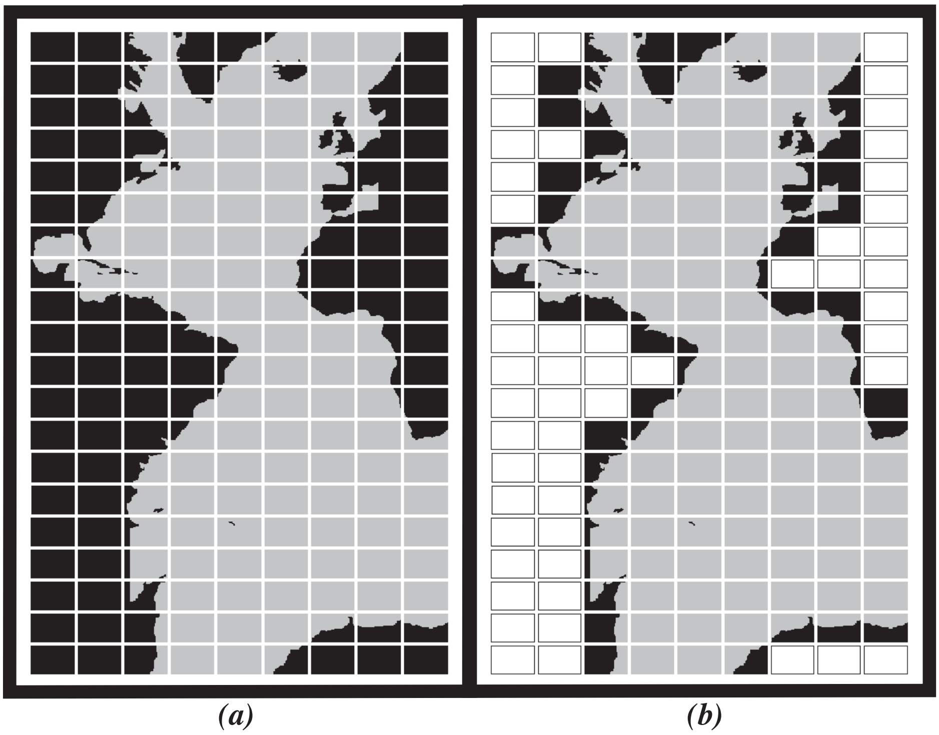

FIG. 7.6 – Example of Atlantic domain defined for the CLIPPER projet. Initial grid is composed by 773 x 1236 horizontal points. (a) the domain is splitting onto 9 subdomains (jpni=9, jpnj=20). 52 subdomains are land areas. (b) 52 subdomains are eliminated (white rectangles) and the resulting number of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

7.4 Open Boundary Conditions (key_obc)

7.5 Flow Relaxation Scheme (? ? ?)

8 Lateral Ocean Physics (LDF)

Contents

The lateral physics on momentum and tracer equations have been given in §2.6.1 and their discrete formulation in §4.2 and §5.5). In this section we further discuss the choices that underlie each lateral physics option. Choosing one lateral physics 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  (+ "key_ldfslp" if ln_sco=T)
   ln_traldf_iso    =  .true.    !     iso-neutral (+ "key_ldfslp")
                                 !  Coefficient
   !  "key_ldftra_c1d"           !     Aht = F(k)
   !  "key_ldftra_c2d"           !     Aht = F(i,j)
   !  "key_ldftra_c3d"           !     Aht = F(i,j,k)
   aht0     =  2000.             !     lateral eddy diffusivity coef. (m2/s)
   ahtb0    =     0.             !     background coef. for isopycnal diffusion (m2/s)
   aeiv0    =  2000.             !     eddy induced velocity coefficient (m2/s)
   !                             !     (+ "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  (+ "key_ldfslp" if ln_sco=T)
   ln_dynldf_iso    =  .false.   !     iso-neutral (+ "key_ldfslp")
   !                             !  Coefficient
   !  "key_ldfdyn_c1d"           !     Ahm = F(k)
   !  "key_ldfdyn_c2d"           !     Ahm = F(i,j)
   !  "key_ldfdyn_c3d"           !     Ahm = F(i,j,k)
   ahm0     = 40000.             !     horizontal eddy viscosity for the dynamics (m2/s)
   ahmb0    =     0.             !     background eddy viscosity for isopycnal diffusion (m2/s)
/
```

8.1 Lateral Mixing Coefficient (key_ldftra_c.d and key_ldfdyn_c.d)

Introducing a space variation in the lateral eddy mixing coefficients changes the model core memory requirement, adding up to four three-dimensional arrays for 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. They allow to specify 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 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 either single or multiple valued functions. Changes in the computer code when switching from one option to another have been minimized by introducing the eddy coefficients as statement function (include file *ldftra_substitute.h90* and *ldfdyn_substitute.h90*). The functions are replaced by their actual meaning during the preprocessing step (cpp capability). The specification of the space variation of the coefficient is settled 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 change these include files following his desiderata. 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 over the whole ocean on momentum and tracers that is specified through *ahm0* and *aht0* namelist parameters.

Vertically varying Mixing Coefficients (key_ldftra_c1d and key_ldfdyn_c1d)

The 1D option is only available in z-coordinate with full step. Indeed in all the other type of vertical coordinate, the depth is a 3D function of (**i,j,j**) and therefore, introducing depth-dependant mixing coefficients will requires 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 is round 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 *ldftra_c1d.h90* file, 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 depend on the local mesh size and the type of operator used :

$$A_l = \begin{cases} \dfrac{\max(e_1, e_2)}{e_{max}} A_o^l & \text{for laplacian operator} \\ \dfrac{\max(e_1, e_2)^3}{e_{max}^3} A_o^l & \text{for bilaplacian operator} \end{cases} \qquad \text{comments} \tag{8.1}$$

where e_{max} is the max of e_1 and e_2 taken over the whole masked ocean domain, and A_o^l is *ahm0* (momentum) or *aht0* (tracer) namelist parameters. 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 [Willebrand et al. 2001].

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 viscous operator uses *ahm0* = $4.10^4 m^2.s^{-1}$ poleward of 20° north and south and decreases to *aht0* = $2.10^3 m^2.s^{-1}$ at the equator [Madec et al. 1996, Holland et al. 2000]. This specification can be found in *ldf_dyn_c2d_orca* routine defined in *ldfdyn_c2d.F90*. Similar specific horizontal variation can be found for Antarctic or Arctic sub-domain 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 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 a ORCA key plus **key_traldf_eiv** plus **key_traldf_c2d** are defined.

A space variation in the eddy coefficient appeals several remarks :

(1) the momentum diffusive 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 a horizontal varying viscosity, the quadratic integral constraints on enstrophy and on the square of the horizontal divergence for operators acting along model-surfaces are no more satisfied (Appendix C).

(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** which default value are 0). Nevertheless, the technique used to compute the isopycnal slopes allows to get rid of such a background diffusion which introduces spurious diapycnal diffusion (see §8.2).

(4) when an eddy induced advection 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 to a biharmonic operator must be set to a *negative* value.

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-coordinate, 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 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 s-coordinate

In s-coordinates, geopotential mixing ($i.e.$ horizontal one) r_1 and r_2 are the slopes between the geopotential and computational surfaces. Their discrete formulation is found by locally vanishing the diffusive fluxes when T is horizontally uniform, i.e. by replacing in (4.11) T by z_T, the depth of T-point, and setting to zero the diffusive fluxes in the three directions. This leads to the following expression for the slopes :

$$\begin{aligned}
r_{1u} &= \frac{e_{3u}}{\left(e_{1u}\ \overline{\overline{e_{3w}}}^{\,i+1/2,\,k}\right)}\ \delta_{i+1/2}[z_T] &&\approx \frac{1}{e_{1u}}\ \delta_{i+1/2}[z_T] \\
r_{2v} &= \frac{e_{3v}}{\left(e_{2v}\ \overline{\overline{e_{3w}}}^{\,j+1/2,\,k}\right)}\ \delta_{j+1/2}[z_T] &&\approx \frac{1}{e_{2v}}\ \delta_{j+1/2}[z_T] \\
r_{1w} &= \frac{1}{e_{1w}}\ \overline{\overline{\delta_{i+1/2}[z_T]}}^{\,i,\,k+1/2} &&\approx \frac{1}{e_{1w}}\ \delta_{i+1/2}[z_{uw}] \\
r_{2w} &= \frac{1}{e_{2w}}\ \overline{\overline{\delta_{j+1/2}[z_T]}}^{\,j,\,k+1/2} &&\approx \frac{1}{e_{2w}}\ \delta_{j+1/2}[z_{vw}]
\end{aligned} \tag{8.2}$$

These slopes are computed once in *ldfslp_init* when *ln_sco*=T and *ln_traldf_hor*=T or *ln_dynldf_hor*=T.

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 to zero diffusive fluxes in the three directions 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) have 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. it 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 McDougall [1987] (see §4.8.2).

z-coordinate with partial step : the technique is identical to the full step case except that at partial step level, the *horizontal* density gradient is evaluated as described in §**??**.

s- or hybrid s-z coordinate : in the current release of *NEMO*, there is no specific treatment for iso-neutral mixing in s-coordinate. In other word, 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 a 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 Griffies [2004], 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 contracion coefficients introduced in §4.8.2, have to be evaluated at the three velocity point. Inorder 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 z and zps 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 OGCM are modified [Weaver and Eby 1997, Griffies et al. 1998]. Here, another strategy has been chosen [Lazar 1997] : 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 diffusive operator (Fig. 8.2.2). This allows an iso-neutral diffusion scheme without additional background horizontal mixing. This technique can be viewed as a diffusive 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 a horizontal background mixing.

Nevertheless, this iso-neutral operator does not ensure that variance cannot increase, contrary to the Griffies et al. [1998] operator which has that property.

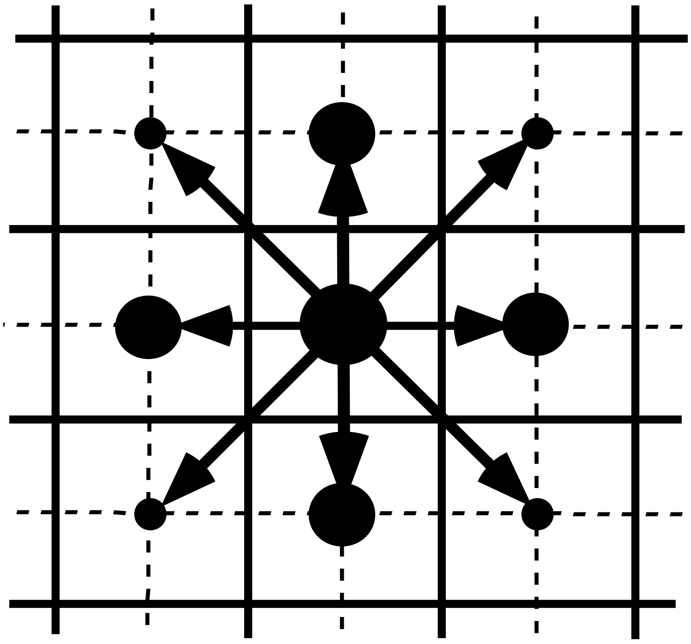

FIG. 8.1 – averaging procedure for isopycnal slope computation.

In addition and also for numerical stability reasons [Cox 1987, Griffies 2004], 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).

add here a discussion about the flattening of the slopes, vs tapering the coefficient.

8.2.3 slopes for momentum iso-neutral mixing

The diffusive iso-neutral operator on momentum is the same as the on used on tracer but applied to each component of the velocity (see (5.25) in section 5.5.2). The slopes between the surface along which the diffusive operator acts and the surface of computation (z- or s-surfaces) are defined at T-, $f-$, and uw-points for the u-component, and $f-T$-, vw-points for the v-component. They are computed as follows from the slopes used for tracer diffusion, i.e. (8.2) and (8.3) :

$$\begin{aligned} r_{1T} &= \overline{r_{1u}}^{\,i} & & & r_{1f} &= \overline{r_{1u}}^{\,i+1/2} \\ r_{2f} &= \overline{r_{2v}}^{\,j+1/2} & & & r_{2T} &= \overline{r_{2v}}^{\,j} \\ r_{1uw} &= \overline{r_{1w}}^{\,i+1/2} & &\text{and} & r_{1vw} &= \overline{r_{1w}}^{\,j+1/2} \\ r_{2uw} &= \overline{r_{2w}}^{\,j+1/2} & & & r_{2vw} &= \overline{r_{2w}}^{\,j+1/2} \end{aligned} \tag{8.6}$$

The major issue remains in the specification of the boundary conditions. The choice made consists in keeping the same boundary conditions as 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 iso-neutral mixing, the slopes use 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 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 the mdltraadv_eiv. This has been preferred to a separate computation of the advective trends associated to the eiv velocity as it allows to take advantage of all the advection schemes offered for the tracers (see §4.1) and not only the 2^{nd} order advection scheme as in previous release of OPA [Madec et al. 1998]. This is particularly useful for passive tracers where *positivity*of the advection scheme is of paramount importance.

At surface, lateral and bottom boundaries, the eddy induced velocity and thus the advective eddy fluxes of heat and salt are set to zero.

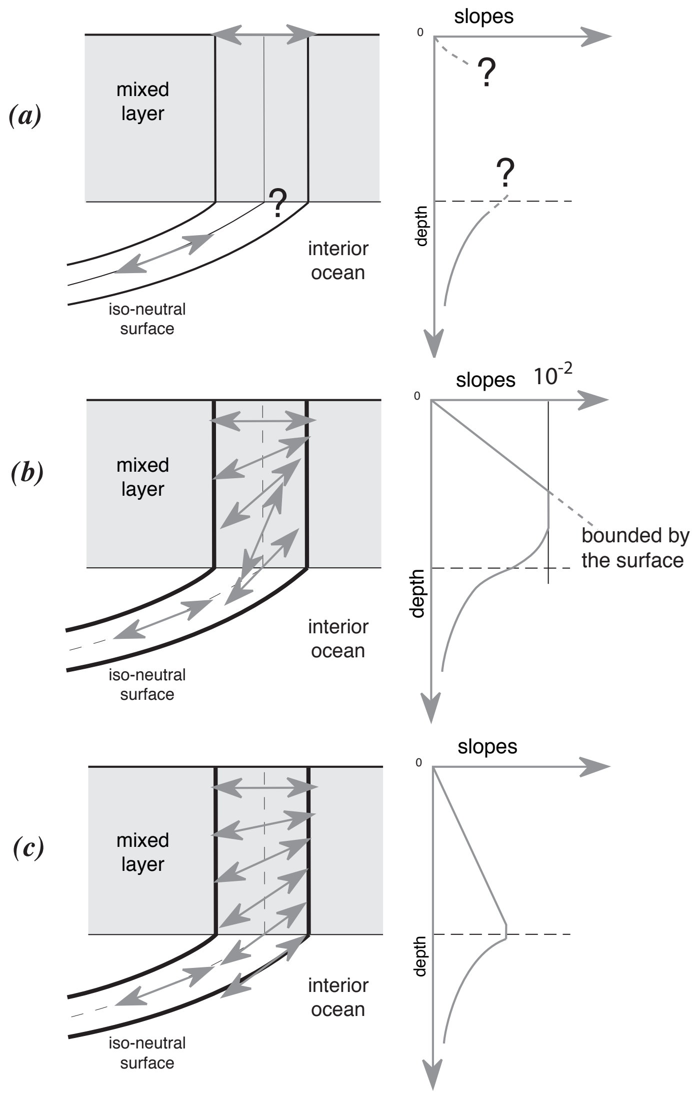

FIG. 8.2 – Vertical profile of the slope used for lateral mixing in the mixed layer : *(a)* in the real ocean the slope is the iso-neutral slope in the ocean interior and their have to adjust to the surface boundary (i.e. tend to zero at the surface as there is no mixing across the air-sea interface : wall boundary condition). Nevertheless, the profile between surface zero value and interior iso-neutral one is unknown, and especially the value at the based 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 actuelly used in *NEMO* : 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 based of the mixed layer between *(b)* and *(c)*. .

9 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 parameterization 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 *zdfini.F90* module and performed in *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 *dynzdf.F90* and *trazdf.F90* modules, respectively. These trends can be computed using either a forward time scheme (cpp variable *np_zdfexp* or a backward time scheme (default option) depending on the magnitude of the mixing coefficients used, and thus of the formulation used (see §3.4).

9.1.1 Constant (key_zdfcst)

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

When the **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 to use this option only in process studies, not in basin scale simulation. 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 *avm0* and *avt0* namelist parameters. In any case, do not use values smaller that those associated to 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"               !  Activate Kz =function of Ri
   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 of the vertical momentum and tracer eddy coefficients is set. The vertical mixing coefficients are diagnosed from the large scale variables computed by the model (order 0.5 closure scheme). *In situ* measurements allow 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 turbulent eddy coefficients and the local Richardson number ($i.e.$ ratio of stratification over vertical shear). Following Pacanowski and Philander [1981], the following formulation has been implemented :

$$\begin{cases} A^{vT} = \dfrac{A^{vT}_{ric}}{(1+a\ Ri)^n} + A^{vT}_b \\ \\ A^{vm} = \dfrac{A^{vT}}{(1+a\ Ri)} + A^{vm}_b \end{cases} \tag{9.1}$$

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 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 coefficients can be modified by setting *avmri*, *alp* and *nric* namelist parameter, respectively.

9.1.3 TKE Turbulent Closure Scheme (key_zdftke)

```
!-----------------------------------------------------------------------
&namtke    !   turbulent eddy kinetic dependent vertical diffusion
!-----------------------------------------------------------------------
!   "key_zdftke"               !  Activate the TKE physics
   ln_rstke = .false.          !  restart with tke from a run without tke
   nitke =         50          !  number of restart iterative loops
   ediff =        0.1          !  coef. for avt (avt=ediff*mxl*sqrt(e))
   ediss =        0.7          !  coef. for Kolmogoroff dissipation
   ebb   =       3.75          !  coef. for surface input of tke
   efave =         1.          !  coef. for enhance Kz on tke (avtke=efave*avm)
   emin  =      1.e-6          !  background value of tke (m^2/s^2)
   emin0 =      1.e-4          !  surface minimum value of tke (m^2/s^2)
!! ri_c  =         0.22222222  !  critic richardson number (default = 2/9)
   nmxl  =          2          !  mixing length type
   !                           !    = 0 bounded by the distance to surface & bottom
   !                           !    = 1 bounded by the local vertical scale factor
   !                           !    = 2 abs( dz(mxl) /e3 ) < 1
   !                           !    = 3 same as 2 with ldiss /= lmix
   npdl  =          1          !  prandtl number
   !                           !    = 0 no vertical prandtl number (avt=avm)
   !                           !    = 1 prandtl number = F(Ri) (avt=pdl*avm)
```

```
  !                       !    = 2 same as =1 but with a shapiro filter on pdl
  nave  =        1        !  horizontal filter on avt and amv (=1) or not (=0)
  navb  =        0        !  constant (=0) or profile (=1) background on avt
/
```

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 Bougeault and Lacarrere [1989] in atmospheric cases, adapted by Gaspar et al. [1990] for oceanic cases, and embedded in OPA by Blanke and Delecluse [1993] for equatorial Atlantic simulations. Since then, significant modifications have been introduced by Madec et al. [1998] 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 Kolmogorov [1942] 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 designates the local Brunt-Vaisälä frequency (see §4.8.2), l_ϵ and l_κ are the dissipation and mixing turbulent 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 [Gaspar et al. 1990]. They are set through namelist parameter *ediff* and *ediss*. P_{rt} can be set to unity or, following Blanke and Delecluse [1993], 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 be optionally applied to R_i. Nevertheless it is an obsolescent option that is notrecommanded. The choice of P_{rt} is controlled by *npdl* namelist parameter.

For computational efficiency, the original formulation of the turbulent length scales proposed by Gaspar et al. [1990] has been simplified. Four formulations are proposed, the choice of which is controlled by *nmxl* namelist parameter. The first two are based on the following first order approximation [Blanke and Delecluse 1993] :

$$l_k = l_\epsilon = \sqrt{2\bar{e}} / N \tag{9.4}$$

which is obtained in a stable stratified region with constant values of the brunt-Vaisälä frequency. The resulting turbulent 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). Blanke and Delecluse [1993] notice that this simplification has two major drawbacks : it has no sense for

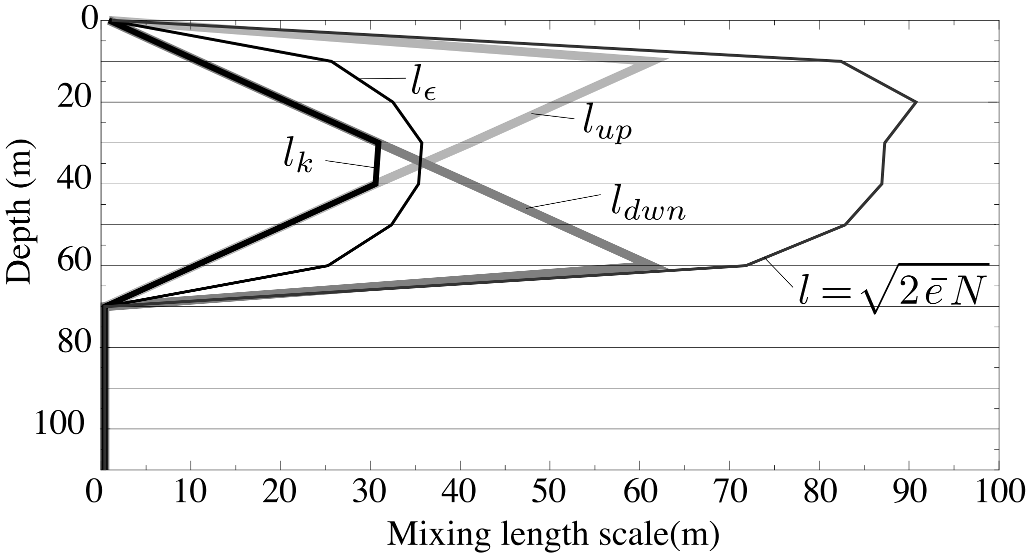

FIG. 9.1 – Illustration of the mixing length computation.

local unstable stratification and the computation no longer uses the whole information contained in the vertical density profile. To overcome this drawbacks, Madec et al. [1998] introduces the *nmxl*=2 or 3 cases, which add of an hypothesis on 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 Gaspar et al. [1990] formulation while being much less time consuming. In particular, it allows the length scale to be limited not only by 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 imposed (9.5) constraint, we introduce two additonnal length scal : l_{up} and l_{dwn}, the upward and downward length scale, and evaluate the dissipation and mixing turbulent length scales as (caution here we use the numerical indexation) :

$$\begin{aligned} l_{up}^{(k)} &= \min\left(l^{(k)}\ ,\ l_{up}^{(k+1)} + e_{3T}^{(k)}\right) && \text{from } k=1 \text{ to } jpk \\ l_{dwn}^{(k)} &= \min\left(l^{(k)}\ ,\ l_{dwn}^{(k-1)} + e_{3T}^{(k-1)}\right) && \text{from } k=jpk \text{ to } 1 \end{aligned} \tag{9.6}$$

where $l^{(k)}$ is compute using (9.4), $i.e.$ $l^{(k)} = \sqrt{2\bar{e}^{(k)}/N^{(k)}}$.

In the *nmxl*=2 case, the dissipation and mixing turbulent length scales take a same value : $l_k = l_\epsilon = \min\left(\ l_{up}\ ,\ l_{dwn}\ \right)$, while in the *nmxl*=2 case, the dissipation and mixing turbulent length scales are give as in Gaspar et al. [1990] :

$$\begin{aligned} l_k &= \sqrt{\ l_{up}\ \ l_{dwn}\ } \\ l_\epsilon &= \min\left(\ l_{up}\ ,\ l_{dwn}\ \right) \end{aligned} \tag{9.7}$$

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 bottom value 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 the positivity. To overcome this problem, a cut-off in the minimum value of $\bar{e}$ is used. Following Gaspar et al. [1990], the cut-off value is set to $\sqrt{2}/2\ 10^{-6}\ m^2.s^{-2}$. This allows the subsequent formulations to match Gargett [1984] one 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"                     !  activate the KPP physics
!   "key_kppcustom"                  !  KPP option 1
!   "key_kpplktb"                    !  KPP option 2
    ln_kpprimix  = .true.            !  shear instability mixing  (default T)
    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 (1/s**2) for maximum convection
    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 KKP scheme has been implemented by J. Chanut ... Add a description of KPP here.

9.2 Convection

```
!-----------------------------------------------------------------------
&namzdf    !   vertical physics
!-----------------------------------------------------------------------
    ln_zdfnpc = .false.              !  Non-Penetrative Convection
    avm0      = 1.2e-4               !  Kz on momemtum (m2/s)
    !                                !  (background Kz if not "key_zdfcst")
    avt0      = 1.2e-5               !  Kz for tracers (m2/s)
    !                                !  (background Kz if not "key_zdfcst")
    ln_zdfevd = .true.               !  enhanced vertical diffusion
    avevd     =   100.               !  Kz for enhanced diffusion scheme (m2/s)
    n_evdm    =     0                !  enhanced mixing Kz apply on tracer (=0)
```

```
   !                          !      or on both tracer and momentum (=1)
   ln_zdfexp =  .false.       !  =T/F  split explicit / implicit
   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 : they must be parameterized. Three parameterisations are available to deal with convective processes : either 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*=T)

```
!-----------------------------------------------------------------------
&namnpc   !   non penetrative convective adjustment
!-----------------------------------------------------------------------
   nnpc1   =      1           !  computation frequency    (time-step)
   nnpc2   =    365           !  control print frequency  (time-step)
/
```

The non-penetrative convective adjustment algorithm is used when *ln_zdfnpc*=T. 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) [Madec et al. 1991a]. This algorithm is an iterative process used in the following way (Fig. 9.2.1) : going from the top of the ocean towards the bottom, the first instability is searched. Assume in the following that the instability is located between levels k and $k+1$. The two levels are vertically mixed, for potential temperature and salinity, 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 two by two statically unstable levels. The latter procedure cannot converge with a finite number of iterations for some vertical profiles while the algorithm used in OPA converges for any profile in a number of iterations less than the number of vertical levels. This property is of paramount importance as pointed out by Killworth [1989] : it avoids the existence of permanent and unrealistic static instabilities at the sea surface. This non-penetrative convective algorithm has been proved successful in studying the deep water formation in the north-western Mediterranean Sea [Madec et al. 1991a;b, Madec and Crépon 1991].

Note that in this algorithm the potential density referenced to the sea surface is used to check whether the density profile is stable or not. Moreover, the mixing in 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. Moreover, temperature and salinity, and

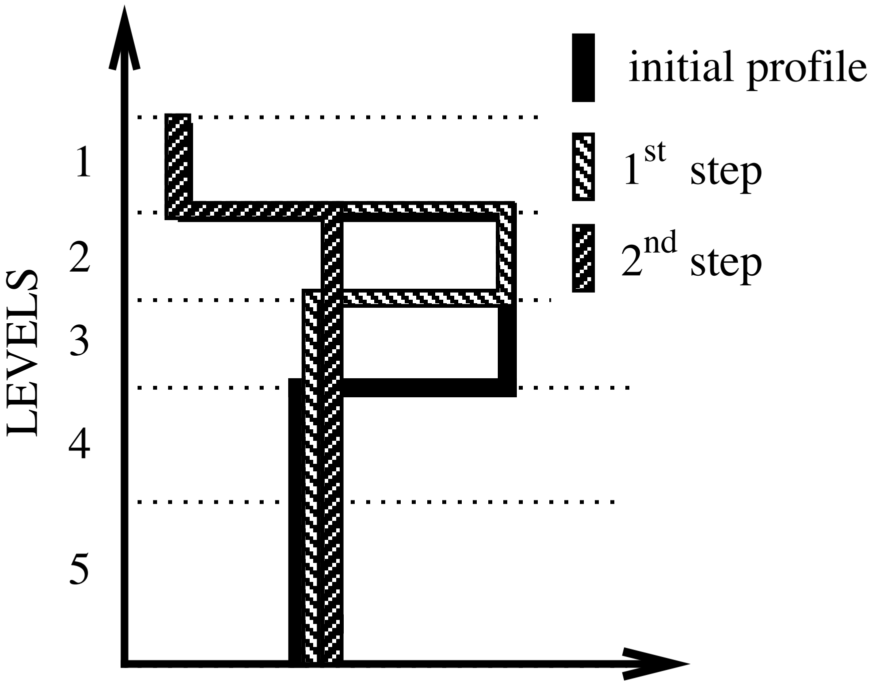

FIG. 9.2 – Example of an unstable density profile treated by the non penetrative convective adjustment algorithm. 1^{st} step : the initial profile is checked from the surface to the bottom. It is found to be unstable between levels 3 and 4. They are mixed. The resulting ρ is still larger than $\rho(5)$: 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.

thus density, are mixed, but the corresponding velocity fields remain unchanged. When using a Richardson 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 algorithm is used with constant vertical eddy viscosity, spurious solution can occur as the vertical momentum diffusion remains small even after a static adjustment. In that latter case, we recommend to add momentum mixing in a manner that mimics the mixing in temperature and salinity [Speich 1992, Speich et al. 1996].

9.2.2 Enhanced Vertical Diffusion (*ln_zdfevd*=T)

```
!-----------------------------------------------------------------------
&namzdf   !   vertical physics
```

```
!-----------------------------------------------------------------------
   ln_zdfnpc = .false.         !  Non-Penetrative Convection
   avm0      = 1.2e-4          !  Kz on momemtum (m2/s)
   !                           !  (background Kz if not "key_zdfcst")
   avt0      = 1.2e-5          !  Kz for tracers (m2/s)
   !                           !  (background Kz if not "key_zdfcst")
   ln_zdfevd = .true.          !  enhanced vertical diffusion
   avevd     =   100.          !  Kz for enhanced diffusion scheme (m2/s)
   n_evdm    =     0           !  enhanced mixing Kz apply on tracer (=0)
   !                           !      or on both tracer and momentum (=1)
   ln_zdfexp =  .false.        !  =T/F  split explicit / implicit
   n_zdfexp  =     3           !  number of sub-timestep for ln_zdfexp=T
/
```

The enhanced vertical diffusion parameterization is used when *ln_zdfevd* is defined. In this case, the vertical eddy mixing coefficients are assigned to be very large (a typical value is 1 $m^2 s^{-1}$) in regions where the stratification is unstable (i.e. when the Brunt-Vaisälä frequency is negative) [Lazar 1997, Lazar et al. 1999]. This is done either on tracers only (*n_evdm*=0) or on both momentum and tracers (*n_evdm*=1) mixing coefficients.

In practice, when $N^2 \leq 10^{-12}$, A^{vT}_T and A^{vS}_T are set to a large value, *avevd*, and if *n_evdm*=1, the four neighbouring A^{vm}_u and A^{vm}_v . Typical value for $avevd$ is in-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 case of static instabilities. It requires the use of an implicit time stepping on vertical diffusion terms (i.e. *ln_zdfexp*=F).

9.2.3 Turbulent Closure Scheme (key_zdftke)

The turbulent closure scheme presented in §9.1.3 and used when the **key_zdftke** is defined allows, in theory, to deal with statically unstable density profiles. In such a case, the term of destruction of turbulent kinetic energy through stratification in (9.2) becomes a source term as N^2 is negative. It results large values of both A^{vT}_T and the four neighbouring$A^{vm}_u and\ A^{vm}_v$ (up to 1 $m^2 s^{-1}$) that are able to restore the static stability of the water column in a way similar to that of the enhanced vertical diffusion parameterization (§9.2.2). Nevertheless, the eddy coefficients computed by the turbulent scheme do usually not exceed $10^{-2} m.s^{-1}$ in the vicinity of the sea surface (first ocean layer) due to the bound of the turbulent length scale by the distance to the sea surface (see §VI.7-c). It can thus be useful to combine the enhanced vertical diffusion with the turbulent closure, i.e. defining *np_zdfevd* and **key_zdftke** CPP variables all together.

The KPP scheme 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 (key_zdfddm)

```
!-----------------------------------------------------------------------
&namddm   !   double diffusive mixing parameterization
!-----------------------------------------------------------------------
!     "key_zdfddm"             !  Activate ddm phisics
```

```
   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 oceans. Merryfield et al. [1999] include a parameterization 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

$$
\begin{aligned}
A^{vT} &= A_o^{vT} + A_f^{vT} + A_d^{vT} \\
A^{vS} &= A_o^{vS} + A_f^{vS} + A_d^{vS}
\end{aligned}
$$

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 buoyancy ratio $R_\rho = \alpha \partial_z T / \partial_z S$, where α and β are coefficients of thermal expansion 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) :

$$
\begin{aligned}
A_f^{vS} &= \begin{cases} \frac{A^{*v}}{1+(R_\rho/R_c)^n} & \text{if } R_\rho > 1 \text{ and } N^2 > 0 \\ 0 & \text{otherwise} \end{cases} \qquad (9.8) \\
A_f^{vT} &= 0.7 \; A_f^{vS} / R_\rho \qquad (9.9)
\end{aligned}
$$

The factor 0.7 in (9.8) reflects the measured ratio $\alpha F_T / \beta F_S \approx 0.7$ of buoyancy fluxes due to transport of heat and salt (e.g., McDougall and Taylor 1984). Following Merryfield et al. [1999], 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 :

$$
A_d^{vT} = \begin{cases} 1.3635 \; \exp\left(4.6 \; \exp\left[-0.54 \left(R_\rho^{-1} - 1\right)\right]\right) & \text{if } 0 < R_\rho < 1 \text{ and } N^2 > 0 \\ 0 & \text{otherwise} \end{cases} \qquad (9.10)
$$

$$
A_d^{vS} = \begin{cases} A_d^{vT} \; (1.85 \, R_\rho - 0.85) & \text{if } 0.5 \leq R_\rho < 1 \text{ and } N^2 > 0 \\ A_d^{vT} \; 0.15 \; R_\rho & \text{if } \; 0 < R_\rho < 0.5 \text{ and } N^2 > 0 \\ 0 & \text{otherwise} \end{cases} \qquad (9.11)
$$

The dependences of (9.8) to (9.10) on R_ρ are illustrated in Fig. 9.3. Implementing this requires computing R_ρ at each grid point and 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).

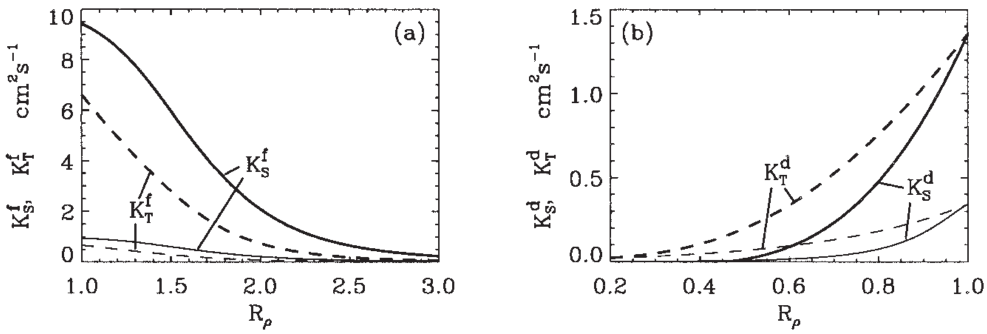

FIG. 9.3 – From Merryfield et al. [1999] : (a) Diapycnal diffusivities A_f^{vT} and A_f^{vS} for temperature and salt in regions of salt fingering. Heavy curves denote $A^{*v} = 10^{-3}\ m^2.s^{-1}$ and thin curves $A^{*v} = 10^{-4}\ m^2.s^{-1}$; (b) diapycnal diffusivities A_d^{vT} and A_d^{vT} for temperature and salt in regions of diffusive convection. Heavy curves denote the Federov parameterization and thin curves the Kelley parameterization. The latter is not implemented in *NEMO*

9.4 Bottom Friction

```
!-----------------------------------------------------------------------
&nambfr    !   bottom friction
!-----------------------------------------------------------------------
   nbotfr =       1        !  type of bottom friction
   !                       !   = 0 : no slip         ,  = 2 : nonlinear friction
                           !   = 1 : linear friction,   = 3 : free slip
   bfri1   =   4.e-4       !  bottom drag coefficient (linear case)
   bfri2   =   1.e-3       !  bottom drag coefficient (non linear case)
   bfeb2   =  2.5e-3       !  bottom tke background (m^2/s^2)
/
```

Both 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 OPA : 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 OPA.

9.4.1 Linear Bottom Friction

(**namelist** !nbotfr : *nbotfr = 0, = 1 or = 3*)

The linear bottom friction parameterization 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 a friction coefficient expressed in m.s^{-1}. This coefficient is generally estimated by setting a typical decay time τ in the deep ocean, $r = H/\tau$. Commonly accepted values of τ are of the order of 100 to 200 days [Weatherly 1984]. A value $\tau^{-1} = 10^{-7}$ s^{-1} corresponding to 115 days is usually used in quasi-geostrophic models. One may consider the linear friction as an approximation of quadratic friction, $r \approx 2\ C_D\ U_{av}$ (Gill [1982], Eq. 9.6.6). With a drag coefficient $C_D = 0.002$, a typical value of tidal currents $U_{av} = 0.1$ 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 OPA. 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 specified 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 $\mathbf{U}_h = 0$ inside the bottom, leads to

$$\begin{aligned} A_u^{vm} &= r\ e_{3uw} \\ A_v^{vm} &= r\ e_{3uw} \end{aligned} \tag{9.15}$$

Such an update is done in *zdfbfr.F90* when *nbotfr*=1 and the value of r used is *bfric1*. Setting *nbotfr*=3 is equivalent to set $r = 0$ and leads to a free-slip bottom boundary condition, while setting *nbotfr*=0 imposes $r = 2\ A_{vb}^{\mathbf{U}}$, where $A_{vb}^{\mathbf{U}}$ is the background vertical eddy coefficient : a no-slip boundary condition is used. Note that this latter choice generally leads to an underestimation of the bottom friction : for 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

(**namelist** !nbotfr : *nbotfr = 2*)

The non-linear bottom friction parameterization 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 $e_b = 2.5 \quad 10^{-3}$m^2.s^{-2}. The FRAM choices have been set as default value (*bfric2* and *bfeb2* namelist parameters).

As for the linear case, the bottom friction is specified in the code by updating the value of the vertical eddy coefficient at the bottom level :

$$\begin{aligned} A_u^{vm} &= C_D \; e_{3uw} \left[u^2 + \left(\bar{\bar{v}}^{i+1,j}\right)^2 + e_b \right]^{1/2} \\ A_v^{vm} &= C_D \; e_{3uw} \left[\left(\bar{\bar{u}}^{i+1,j}\right)^2 + v^2 + e_b \right]^{1/2} \end{aligned} \tag{9.17}$$

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*).

10 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 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)

```
!-----------------------------------------------------------------------
&namcro   !   cross land advection
!-----------------------------------------------------------------------
!  n_cla   advection between 2 ocean pts separates by land
   n_cla   = 1
/
```

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 zoom option (improperly because it is not associated with a change in model resolution) is a quite simple function that allows to perform a simulation over a sub-domain of an already defined configuration (i.e. without defining a new set of mesh, initial state and forcings). This option can be useful for testing the user setting of surface boundary conditions, or the initial ocean state of a huge ocean model configuration while having a small central memory requirement. It can also be used to easily test specific physics in a sub-domain (for example, test of the coupling between sea-ice and ocean model over the Arctic or Antarctic ocean as they are set in the global ocean version of OPA [Madec et al. 1996]. In standard, this option does not include any specific treatment for 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 those 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 configuration, the user set *jpiglo=jpidta jpjglo=jpjdta* and *jpk=jpkdta*. When running the model over a sub-domain, the user have 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).

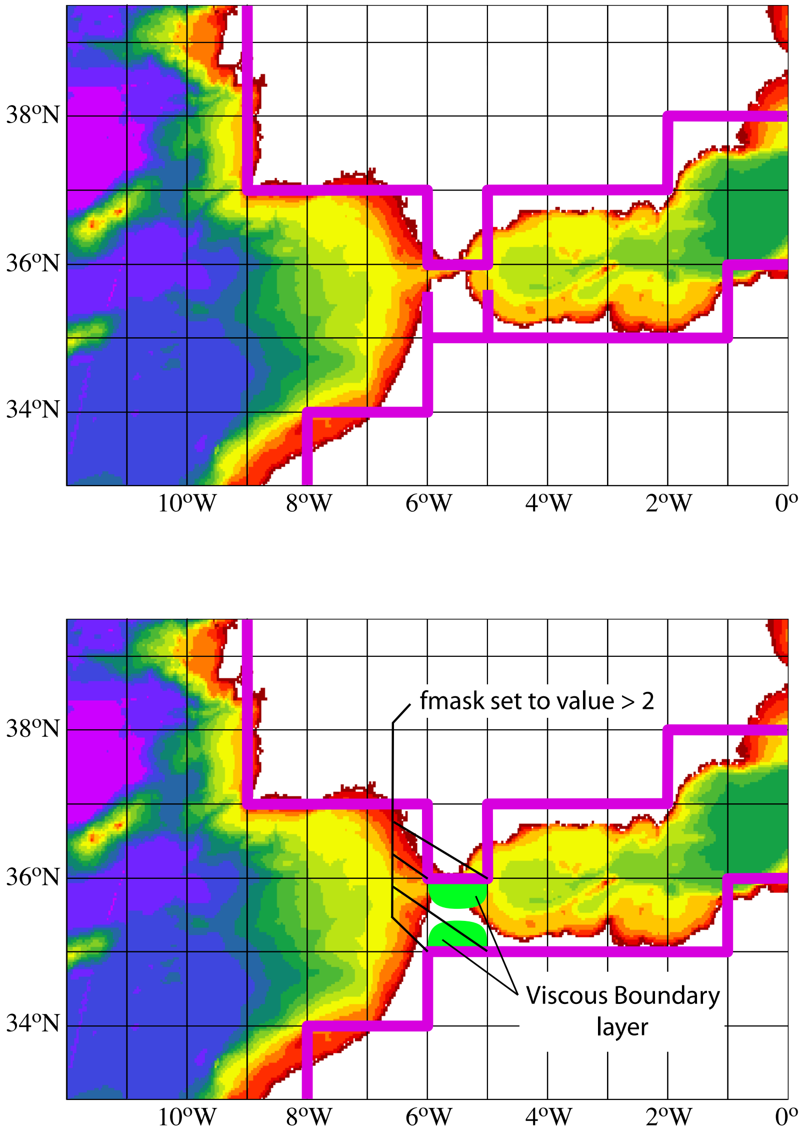

FIG. 10.1 – Example of the Gibraltar strait defined in a 1° x 1° mesh. *Top* : using partially open cells. The meridional scale factor at V-point is reduced on both sides of the strait to account for the real width of the strait (about 20 km). Note that the scale factors of the strait T-point remains unchanged. *Bottom* : using viscous boundary layers. The four fmask 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 dimension 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 (see §IV.2-c)

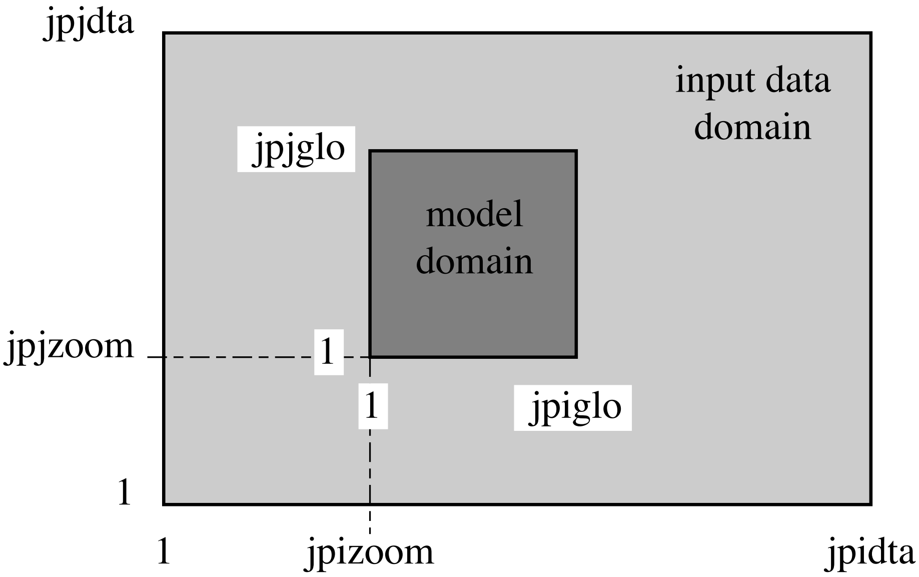

FIG. 10.2 – position of a model domain compared to the data input domain when the zoom functionality is used.

10.4 Water column model : 1D model (key_cfg_1d)

The 1D model is a stand alone water column based on 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 parameterizations of unresolved turbulence (wind steering, langmuir circulation, skin layers) ; *(c)* to compare the behaviour of different mixing vertical scheme ; *(d)* to perform sensitivity study to the vertical diffusion on a particular point of the ocean global domain ; *(d)* to access to specific diagnostics, aside from the standard model variables, because of having small in core memory requirement.

The methodology is based on the use of the zoom functionality (see §10.3), and the addition of some specific routines. There is no need of defining a new set of mesh, bathymetry, initial state and forcing, as 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       !  = 1 read the bathymetry_level
   e3zps_min =     5.      !  minimum thickness of the partial step is the min of
   e3zps_rat =     0.1     !  e3zps_min and e3zps_rat * e3t  (with 0<e3zps_rat<1)
   nmsh      =     0       !  =1 create a mesh file (coordinates, scale factors, masks)
   nacc      =     0       !  the acceleration of convergence method
   !                       !     = 0, no acceleration, rdt = rdttra
   !                       !     = 1, acceleration used, rdt < rdttra(k)
   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
   rdtmax    =  5760.      !  maximum time step on tracers
   rdth      =   800.      !  depth variation of tracer time step
   rdtbt     =    90.      !  barotropic time step (for the time splitting algorithm)
   nfice     =     5       !  frequency of ice model call
   nfbulk    =     5       !  frequency of bulk formulea call (not used if ice used)
   nclosea   =     0       !  = 0 no closed sea in the model domain
!                          !  = 1 closed 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 consideration (less than a few hours), this is usually requires a large elapse time. In order overcome this problem, Bryan [1984] introduces a technique that allows to accelerate the spin up to the equilibrium. It consists in using 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 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. Both are settled from *rdt* and *rdttra* namelist parameters. The set of prognostic equations to solve becomes :

$$
\begin{aligned}
\frac{\partial \mathbf{U}_h}{\partial t} &\equiv \frac{\mathbf{U}_h^{t+1} - \mathbf{U}_h^{t-1}}{2\Delta t} = \ldots \\
\frac{\partial T}{\partial t} &\equiv \frac{T^{t+1} - T^{t-1}}{2\widetilde{\Delta t}} = \ldots \\
\frac{\partial S}{\partial t} &\equiv \frac{S^{t+1} - S^{t-1}}{2\widetilde{\Delta t}} = \ldots
\end{aligned}
\qquad (10.1)
$$

Bryan [1984] has analysed 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 there is a wider range of instability. This technique is efficient for searching 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 keep this technique when using 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 $\Delta\tilde{t}$ implies that the heat and salt contents are no more conserved due to the vertical coupling of the ocean level through both advection and diffusion.

10.6 Model optimisation, Control Print and Benchmark

```
!-----------------------------------------------------------------------
&nam_ctl      !   Control prints & Benchmark
!-----------------------------------------------------------------------
   ln_ctl =  .false.        !  trends control print (expensive!)
   nprint =       0         !  level of print (0 no print)
   nictls =       0         !  start i indice to make the control SUM (very usefull to compare mono-
   nictle =       0         !  end   i indice to make the control SUM (-versus multi processor runs)
   njctls =       0         !  start j indice to make the control SUM (very usefull to compare mono-
   njctle =       0         !  end   j indice to make the control SUM (-versus multi processor runs)
   isplt  =       1         !  number of processors following i
   jsplt  =       1         !  number of processors following j
   nbench =       0         !  = 1 Bench run (no physical meaning)
                            !  = 0 Standard simulation
   nbit_cmp =     0         !  = 1 enables bit comparison between mono and mpp runs
                            !  = 0 faster mpp run
/
```

Three points to be described here :

• Vector and memory optimisation :

key_vectopt_loop enable the internal loop collapse, a very efficient way to increase the length of vector and thus speed up the model on vector computers.

Add here also one word on NPROMA technique that has been found useless, since compiler have made significant progress during the last decade.

Add also one word on NEC specific optimisation (Novercheck option for example)

key_vectopt_memory has been introduced in order to reduce the memory requirement of the model. This is obviously done by incresing the CPU time requirement, as it suppress intermediate computation saved in in-core memory. This possibility have not been intensively used. In fact up to now, it only concern the TKE physics, in which, when **key_vectopt_memory** is defined, the coefficient used for horizontal smoothing of A_v^T and A_v^m are no more computed once for all. This reduces the memory requirement by three 2D arrays.

• Control print : describe here 4 things :

1- *ln_ctl* : compute and print the inner domain averaged trends in all TRA, DYN LDF and ZDF modules. Very useful to detect the origin of an undesired change in model results.

2- also *ln_ctl* but using the nictl. njctl. namelist parameters to check the origin of differences between mono and multi processor

3- **key_esopa** (to be rename key_nemo) : also a option of model management. When defined, this key force the activation of all options and CPP keys. For example, all the tracer and momentum advection scheme are called ! There is therefore no physical meaning associated with model results. In fact, this option allows both the compilator and the model run to go through all the Fortran lines of the model. This allows to check is there is obvious compilation or running bugs for CPP options, and running bugs for namelist options.

5- last digit comparison (*nbit_cmp*). In MPP simulation, the computation of sum of the whole domain is performed as the sum over all processors of the sum of the inner domain of each processor. This double sum nevr give exactly the same result as a single sum over the whole domain, due to truncature differences. The ”bit comparison” option has been introduced in order to be able to check that mono-processor and multi-processor give exactly the same results.

• Benchmark (*nbench*). This option, defines a benchmark run based on GYRE configuration in which the resolution remains the same whatever the domain size is. This allows to run very large model domain by just changing the domain size (*jpiglo*, *jpjglo*) without adjusting neither the time-step nor the physical parametrisations.

10.7 Elliptic solvers (SOL)

```
!-----------------------------------------------------------------------
&namsol   !   elliptic solver / island / free surface
!-----------------------------------------------------------------------
   nsolv     =      1         !  type of 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
   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 ("key_dynspg_flt)
/
```

The computation of the surface pressure gradient with a rigid lid assumption requires to compute $\partial_t \psi$, the time evolution of the barotropic streamfunction. $\partial_t \psi$ is solution of an elliptic equation (I.2.4) for which two solvers are available, a Successive-Over-Relaxation (SOR) or a preconditioned conjugate gradient (PCG) [Madec et al. 1988, Madec 1990]. The PCG is a very efficient method for solving elliptic equations on vector computers. It is a fast and rather easy to use method, which is an attractive feature 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 the search of an optimal parameter as in the SOR method. Nevertheless, the SOR has been kept because it is a linear solver, a very useful property when using the adjoint model of OPA.

The surface pressure gradient is computed in *dynspg.F90*. The default option is the use of PCG or SOR depending on *nsolv* (namelist parameter). At each time step the time derivative of the barotropic streamfunction is the solution of (II.2.3). Introducing the following coefficients :

$$
\begin{aligned}
C_{i,j}^{NS} &= \frac{e_{2v}(i,j)}{(H_v(i,j)e_{1v}(i,j))} \\
C_{i,j}^{EW} &= \frac{e_{1u}(i,j)}{(H_u(i,j)e_{2u}(i,j))} \\
B_{i,j} &= \delta_i \left(e_{2v} M_v\right) - \delta_j \left(e_{1u} M_u\right)
\end{aligned}
\tag{10.2}
$$

the five-point finite difference equivalent equation (II.2.3) can be rewritten as :

$$\begin{split} C^{NS}_{i+1,j}\left(\frac{\partial\psi}{\partial t}\right)_{i+1,j} + C^{EW}_{i,j+1}\left(\frac{\partial\psi}{\partial t}\right)_{i,j+1} + C^{NS}_{i,j}\left(\frac{\partial\psi}{\partial t}\right)_{i-1,j} + C^{EW}_{i,j}\left(\frac{\partial\psi}{\partial t}\right)_{i,j-1} \\ -\left(C^{NS}_{i+1,j} + C^{EW}_{i,j+1} + C^{NS}_{i,j} + C^{EW}_{i,j}\right)\left(\frac{\partial\psi}{\partial t}\right)_{i,j} = B_{i,j} \end{split} \tag{10.3}$$

(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 :

$$\begin{split} A^N_{i,j}\left(\frac{\partial\psi}{\partial t}\right)_{i+1,j} + A^E_{i,j}\left(\frac{\partial\psi}{\partial t}\right)_{i,j+1} \\ + A^S_{i,j}\left(\frac{\partial\psi}{\partial t}\right)_{i-1,j} + A^W_{i,j}\left(\frac{\partial\psi}{\partial t}\right)_{i,j-1} - \left(\frac{\partial\psi}{\partial t}\right)_{i,j} = \tilde{B}_{i,j} \end{split} \tag{10.4}$$

The SOR method used is an iterative method. Its algorithm can be summarised as follows [see Haltiner and Williams [1980] for further discussion] :

initialisation (evaluate a first guess from former time step computations)

$$\left(\frac{\partial\psi}{\partial t}\right)^0_{i,j} = 2\left(\frac{\partial\psi}{\partial t}\right)^t_{i,j} - \left(\frac{\partial\psi}{\partial t}\right)^{t-1}_{i,j} \tag{10.5}$$

iteration n, from $n = 0$until convergence, do :

$$\begin{split} R^n_{i,j} = A^N_{i,j}\left(\frac{\partial\psi}{\partial t}\right)^n_{i+1,j} + A^E_{i,j}\left(\frac{\partial\psi}{\partial t}\right)^n_{i,j+1} \\ + A^S_{i,j}\left(\frac{\partial\psi}{\partial t}\right)^{n+1}_{i-1,j} + A^W_{i,j}\left(\frac{\partial\psi}{\partial t}\right)^{n+1}_{i,j-1} - \left(\frac{\partial\psi}{\partial t}\right)^n_{i,j} - \tilde{B}_{i,j} \end{split} \tag{10.6}$$

$$\left(\frac{\partial\psi}{\partial t}\right)^{n+1}_{i,j} = \left(\frac{\partial\psi}{\partial t}\right)^n_{i,j} + \omega\, R^n_{i,j} \tag{10.7}$$

where ω satisfies $1 \leq \omega \leq 2$. An optimal value exists for ω which accelerates significantly the convergence, but it has to be adjusted empirically for each model domain, except for an uniform grid where an analytical expression for ω can be found [Richtmyer and Morton 1967]. This parameter is defined as *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 highly recommended to use a ϵ smaller or equal to 10^{-2} as larger values may leads to numerically induced basin scale barotropic oscillations, and to use a two or three order of magnitude smaller value in eddy resolving configuration. The precision of the solver is not only a numerical parameter, but influences the physics. Indeed, the zero change of kinetic energy due to the work of surface pressure forces in rigid-lid approximation is only achieved at the precision demanded on the solver (§ C.1-e). The precision is specified by setting *eps* (**namelist parameter**). In addition, two other tests are used to stop the iterative algorithm. They concern the number of iterations and the module 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 while 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 (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 for $\epsilon = 10^{-2}$, increasing to a few thousand for $\epsilon = 10^{-12}$.

The vectorization of the SOR algorithm is not straightforward. (VII.5.4) contains two linear recurrences on i and j. This inhibits the vectorisation (§ IV.2-a). Therefore (VII.5.4a) has been rewritten as :

$$\begin{aligned} R^n_{i,j} = A^N_{i,j} \left(\frac{\partial \psi}{\partial t} \right)^n_{i+1,j} &+ A^E_{i,j} \left(\frac{\partial \psi}{\partial t} \right)^n_{i,j+1} \\ &+ A^S_{i,j} \left(\frac{\partial \psi}{\partial t} \right)^n_{i-1,j} + A^W_{i,j} \left(\frac{\partial \psi}{\partial t} \right)^n_{i,j-1} - \left(\frac{\partial \psi}{\partial t} \right)^n_{i,j} - \tilde{B}_{i,j} \end{aligned} \tag{10.9}$$

$$R^n_{i,j} = R^n_{i,j} - \omega \; A^S_{i,j} \; R^n_{i,j-1} \tag{10.10}$$

$$R^n_{i,j} = R^n_{i,j} - \omega \; A^W_{i,j} \; R^n_{i-1,j} \tag{10.11}$$

If the three equations in (VII.5.6) are solved inside the same do-loop, (VII.5.4a) and (VII.5.6) are strictly equivalent. In the model they are solved successively over the whole domain. The convergence is slower but (VII.5.6a) and (VII.5.6b) are vector loops on i-index (the inner loop) and (VII.5.6c) is adapted to Cray vector computers by using a routine from the Cray library (namely the FOLR routine) to solve the first-order linear recurrence. 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 equations.

10.7.2 Preconditioned Conjugate Gradient

(*nbsfs=1*, **namelist parameter**)

A is a definite positive symmetric matrix, thus solving the linear system (VII.5.1) 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 the solution in the following iterative way : assuming that $\mathbf{x}^n$ is obtained, $\mathbf{x}^{n+1}$ is searched under the form $\mathbf{x}^{n+1} = \mathbf{x}^n + \alpha^n \mathbf{d}^n$ which satisfies :

$$\mathbf{x}^{n+1} = \inf_{\mathbf{y}=\,\mathbf{x}^n+\alpha\ \mathbf{d}^n} \phi(\ \mathbf{y}) \quad \Leftrightarrow \quad \frac{d\phi}{d\alpha} = 0$$

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 choice of the descent vector $\mathbf{d}^n$ depends 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 base 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, both the PCG algorithm and the whole model computation are stopped when the number of iteration, $nmax$, or the module of the right hand side exceeds a specified value (see § VII.5-a for further discussion). The precision and the maximum number of iteration are specified by setting eps and $nmax$ (**namelist parameters**).

It can be demonstrated that the 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 all the more fast as the matrix is closed to identity (i.e. the eigen values are closed 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 (VII.5.7) 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** [Madec et al. 1988]. In OPA, **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 (VII.5.8) is in fact given by (VII.5.2) and thus the matrix and right hand side are computed independently from the solver used.

10.7.3 FETI

FETI is a powerfull solver that was developed by Marc Guyon (ref ! ! !). It as been just converted from Fortan 77 to 90, but never successfully tested after that. Since then, nobody has found enough time to further investigate the implmenntationof FETI and debug it.

Its main advantaged is to save a lot of CPU time compared to SOR or PCG algorithm. Nevertheless, its main drawbacks is that the solution is depends 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 defined)

The boundary condition used for both 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 done to determine the barotropic streamfunction with the correct boundary conditions. This is detailed below.

The model does not recognise itself the islands which must be defined using bathymetry information, i.e. $mbathy$ array, equals -1 over the first island, -2 over the second, ... , $-N$ over the N^{th} island (see § VII.2-b). The model determines the position of island grid-points and defines a closed contour around each island which will be used to compute the circulation around each island. The closed contour is formed of the ocean grid-points which are the closest to the island.

First, the island barotropic streamfunctions ψ_n are computed using the PCG (they are solutions of (VII.5.1) with the right-hand side equals 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 define a specific right hand side for (VII.5.1)). The precision of this computation can be very high since it is done once. The absolute precision, $epsisl$, and the maximum number of iteration, $nmisl$, are the **namelist parameters** used for that computation. Their typical values are $epsisl = 10^{-10}$ and $nmisl = 4000$. Then the island matrix A is computed from (I.2.8) and reversed. At each time step, ψ_0, the solution of (I.2.4) with $\psi_0 = 0$ along all coastlines, is computed using either SOR or PCG. It has to be noted that the first guess of this computation is defined as in (VII.5.3) except that $\partial_t \psi_0$ is used, not $\partial_t \psi$. Indeed we are computing $\partial_t \psi_0$ which is usually far different from $\partial_t \psi$. Then, it is easy to find the time evolution of the barotropic streamfunction on each island and to deduce $\partial_t \psi$ using (I.2.9) in order 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)

10.8.2 Tracer/Dynamics Trends (key_trdlmd, key_diatrdtra)

When **key_diatrddyn** and/or **key_diatrddyn** cpp variables are defined, each trends of the dynamics and/or temperature and salinity time evolution equations are stored in three-dimensional arrays just after their computation (i.e. at the end of each $dyn \cdots .F$ and/or $tra \cdots .F$ routines). These trends are then used in diagnostic routines $diadyn.F$ and $diatra.F$ respectively. In standard, these routines check the basin averaged properties of the momentum and tracer equations every *ntrd* time-steps (**namelist parameter**). These routines are given as an example ; they must be adapted by the user to his/her desiderata.

These two options imply the definition of several arrays in the in core memory, increasing quite sensitively the code memory requirements (search for **key_diatrddyn** or **key_diatrdtra**in *common.h* file)

10.8.3 On-line Floats trajectories

```
!-----------------------------------------------------------------------
&namflo    !   float parameters                                        ("key_float")
!-----------------------------------------------------------------------
    ln_rstflo = .false.          !  boolean term for float restart (true or false)
    nwritefl  =      75          !  frequency of float output file
    nstockfl  =    5475          !  frequency of float restart file
    ln_argo   = .false.          !  Argo type floats (stay at the surface each 10 days)
    ln_flork4 = .false.          !  = T trajectories computed with a 4th order Runge-Kutta
                                 !  = F  (default)   computed with Blanke' scheme
/
```

a description is to be added here

10.8.4 Other Diagnostics

Aside from the standard model variables, some other diagnostics are computed online or can be added in the model. The available ready-to-add diagnostics can be found in DIA. Among the available diagnostics one can quote :

- 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*)

Index

Bibliographie

Arakawa, A. and Y.-J. G. Hsu, 1990 : Energy conserving and potential-enstrophy dissipating schemes for the shallow water equations. *Mon. Wea. Rev.*, **118 (10)**, 1960–1969.

Arakawa, A. and V. R. Lamb, 1981 : A potential enstrophy and energy conserving scheme for the shallow water equations. *Mon. Wea. Rev.*, **109 (1)**, 18–36.

Asselin, R., 1972 : Frequency filter for time integrations. *Mon. Wea. Rev.*, **100 (6)**, 487–490.

Beckmann, A. and R. Döscher, 1998 : A method for improved representation of dense water spreading over topography in geopotential-coordinate models. *J. Phys. Oceanogr.*, **27**, 581–591.

Beckmann, A. and D. B. Haidvogel, 1993 : Numerical simulation of flow around a tall isolated seamount. part i - problem formulation and model accuracy. *Journal of Physical Oceanography*, **23 (8)**, 1736–1753.

Blanke, B. and P. Delecluse, 1993 : Low frequency variability of the tropical atlantic ocean simulated by a general circulation model with mixed layer physics. *J. Phys. Oceanogr.*, **23**, 1363–1388.

Bougeault, P. and P. Lacarrere, 1989 : Parameterization of orography-induced turbulence in a mesobeta–scale model. *Mon. Wea. Rev.*, **117 (8)**, 1872–1890.

Brown, J. A. and K. A. Campana, 1978 : An economical time-differencing system for numerical weather prediction. *Mon. Wea. Rev.*, **106 (8)**, 1125–1136.

Bryan, K., 1984 : Accelerating the convergence to equilibrium of ocean-climate models. *J. Phys. Oceanogr.*, **14**.

Bryden, H. L., 1973 : New polynomials for thermal expansion, adiabatic temperature gradient and potential temperature of sea water. *Deep-Sea Res.*, **20**, 401–408.

Campin, J.-M., A. Adcroft, C. Hill, and J. Marshall, 2004 : Conservation of properties in a free-surface model. *Ocean Modelling*, **6, 3-4**, 221–244.

Cox, M., 1987 : Isopycnal diffusion in a z-coordinate ocean model. *Ocean Modelling*, **74**, 1–5.

Dukowicz, J. K. and R. D. Smith, 1994 : Implicit free-surface method for the bryan-cox-semtner ocean model. *J. Geophys. Res*, **99**, 7991–8014.

Eiseman, P. R. and A. P. Stone, 1980 : Conservation lows of fluid dynamics – a survey. *SIAM Review*, **22**, 12–27.

Farge, M., 1987 : Dynamique non lineaire des ondes et des tourbillons dans les equations de saint venant. Ph.D. thesis, Doctorat es Mathematiques, Paris VI University, 401 pp.

Farrow, D. E. and D. P. Stevens, 1995 : A new tracer advection scheme for bryan–cox type ocean general circulation models. *J. Phys. Oceanogr.*, **25**, 1731–1741.

Fujio, S. and N. Imasato, 1991 : Diagnostic calculation for circulation and water mass movement in the deep pacific. *J. Geophys. Res*, **96**, 759–774.

Gargett, A. E., 1984 : Vertical eddy diffusivity in the ocean interior. *J. Mar. Res.*, **42**.

Gaspar, P., Y. Grégoris, and J.-M. Lefevre, 1990 : A simple eddy kinetic energy model for simulations of the oceanic vertical mixing Tests at station papa and long-term upper ocean study site. *J. Geophys. Res*, **95(C9)**.

Gent, P. R. and J. C. Mcwilliams, 1990 : Isopycnal mixing in ocean circulation models. *J. Phys. Oceanogr.*, **20 (1)**, 150–155.

Gill, A. E., 1982 : *Atmosphere-Ocean Dynamics*. International Geophysics Series, Academic Press, New-York.

Griffies, S. M., 2004 : *Fundamentals of ocean climate models*. Princeton University Press, 434pp.

Griffies, S. M., A. Gnanadesikan, R. C. Pacanowski, V. D. Larichev, J. K. Dukowicz, and R. D. Smith, 1998 : Isoneutral diffusion in a z-coordinate ocean model. *J. Phys. Oceanogr.*, **28 (5)**, 805–830.

Griffies, S. M., R. C. Pacanowski, M. Schmidt, and V. Balaji, 2001 : Tracer conservation with an explicit free surface method for z-coordinate ocean models. *Mon. Wea. Rev.*, **129 (5)**, 1081–1098.

Guilyardi, E., G. Madec, and L. Terray, 2001 : The role of lateral ocean physics in the upper ocean thermal balance of a coupled ocean-atmosphere gcm. *Clim. Dyn.*, **17 (8)**, 589–599.

Haltiner, G. J. and R. T. Williams, 1980 : *Numerical prediction and dynamic meteorology*. John Wiley & Sons Eds., second edition, 477pp.

Haney, R. L., 1991 : On the pressure gradient force over steep topography in sigma coordinate ocean models. *J. Phys. Oceanogr.*, **21 (4)**, 610–619.

Holland, W. R., S. Jaussaume, and F. David, (Eds.), 2000 : *Modeling the Earth's Climate and its Variability, Les Houches, Session LXVII 1997*, chap. Ocean modelling and the role of the ocean in the climate system, 237–313.

Jackett, D. R. and T. J. McDougall, 1995 : Minimal adjustment of hydrographic data to achieve static stability. *J. Atmos. Ocean Tech.*, **12**, 381–389.

Killworth, P. D., 1989 : On the parameterization of deep convection in ocean models. *Parameterization of small-scale processes*, winter workshop, H., Ed., University of Hawaii at Manoa.

Killworth, P. D., D. Stainforth, D. J. Webb, and S. M. Paterson, 1991 : The development of a free-surface bryan-cox-semtner ocean model. *J. Phys. Oceanogr.*, **21 (9)**, 1333–1348.

Kolmogorov, A. N., 1942 : The equation of turbulent motion in an incompressible fluid. *Izv. Akad. Nauk SSSR, Ser. Fiz.*, **6**, 56–58.

Large, W. and S. Yeager, 2004 : *Diurnal to decadal global forcing for ocean and sea-ice models : the data sets and flux climatologies*. NCAR Technical Note, NCAR/TN-460+STR, CGD Division of the National Center for Atmospheric Research.

Large, W. G., J. C. McWilliams, and S. C. Doney, 1994 : Oceanic vertical mixing - a review and a model with a nonlocal boundary layer parameterization. *Reviews of Geophysics*, **32**, 363–404, doi :10.1029/94RG01872.

Lazar, A., 1997 : La branche froide de la circulation thermohaline - sensibilité à la diffusion turbulente dans un modèle de circulation générale idéalisée. Ph.D. thesis, Université Pierre et Marie Curie, Paris, France, 200pp.

Lazar, A., G. Madec, and P. Delecluse, 1999 : The deep interior downwelling, the veronis effect, and mesoscale tracer transport parameterizations in an ogcm. *J. Phys. Oceanogr.*, **29 (11)**, 2945–2961.

Leonard, B. P., 1979 : A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Computer Methods in Applied Mechanics and Engineering*, **19**, 59–98.

———, 1991 : The ultimate conservative difference scheme applied to unsteady one–dimensional advection. *Computer Methods in Applied Mechanics and Engineering*, 17–74.

Levier, B., A.-M. Tréguier, G. Madec, and V. Garnier, 2007 : Free surface and variable volume in the nemo code. Tech. rep., MERSEA MERSEA IP report WP09-CNRS-STR-03-1A, 47pp, available on the NEMO web site.

Lévy, M., A. Estubier, and G. Madec, 2001 : Choice of an advection scheme for biogeochemical models. *Geophys. Res. Let.*, **28**.

Madec, G., 1990 : La formation d'eau profonde et son impact sur la circulation régionale en méditerranée occidentale - une approche numérique. Ph.D. thesis, UniversitéPierre et Marie Curie, Paris, France, 194pp.

Madec, G., M. Chartier, and M. Crépon, 1991a : Effect of thermohaline forcing variability on deep water formation in the northwestern mediterranean sea - a high resulution three-dimensional study. *Dyn. Atmos. Ocean.*

Madec, G., M. Chartier, P. Delecluse, and M. Crépon, 1991b : A three-dimensional numerical study of deep water formation in the northwestern mediterranean sea . *J. Phys. Oceanogr.*, **21**.

Madec, G. and M. Crépon, 1991 : *Deep convection and deep water formation in the oceans*, chap. Thermohaline-driven deep water formation in the Northwestern Mediterranean Sea. Elsevier Oceanographic Series.

Madec, G. and P. Delecluse, 1997 : The opa/arpege and opa/lmd global ocean-atmosphere coupled model. *Int. WOCE Newsletter*, **26**, 12–15.

Madec, G., P. Delecluse, M. Imbard, and C. Levy, 1998 : Opa 8 ocean general circulation model - reference manual. Tech. rep., LODYC/IPSL Note 11.

Madec, G. and M. Imbard, 1996 : A global ocean mesh to overcome the north pole singularity. *Clim. Dyn.*, **12**, 381–388.

Madec, G., F. Lott, P. Delecluse, and M. Crépon, 1996 : Large-scale preconditioning of deep-water formation in the northwestern mediterranean sea. *J. Phys. Oceanogr.*, **26 (8)**, 1393–1408.

Madec, G., C. Rahier, and M. Chartier, 1988 : A comparison of two-dimensional elliptic solvers for the barotropic streamfunction in a multilevel ogcm. *Ocean Modelling*, **78**.

Maltrud, M. E., R. D. Smith, A. J. Semtner, and R. C. Malone, 1998 : Global eddy-resolving ocean simulations driven by 1985-1995 atmospheric winds. *J. Geophys. Res*, **103(C13)**, 30,825–30,854.

Marti, O., 1992 : Etude de l'océan mondial : modélisation de la circulation et du transport de traceurs anthropogéniques. Ph.D. thesis, Université Pierre et Marie Curie, Paris, France, 201pp.

Marti, O., G. Madec, and P. Delecluse, 1992 : Comment on "net diffusivity in ocean general circulation models with nonuniform grids" by f. l. yin and i. y. fung. *J. Geophys. Res*, **97**, 12 763–12 766.

McDougall, T. J., 1987 : Neutral surfaces. *Journal of Physical Oceanography*, **17 (11)**, 1950–1964.

Merryfield, W. J., G. Holloway, and A. E. Gargett, 1999 : A global ocean model with double-diffusive mixing. *J. Phys. Oceanogr.*, **29 (6)**, 1124–1142.

Murray, R. J., 1996 : Explicit generation of orthogonal grids for ocean models. *J. Comput. Phys.*, **126 (2)**, 251–273.

Olivier, F., 2001 : Etude de l'activité biologique et de la circulation océanique dans un jet géostrophique : le front alméria-oran. Ph.D. thesis, Université Pierre et Marie Curie, Paris, France.

Pacanowski, R. and S. Philander, 1981 : Parameterization of vertical mixing in numerical models of tropical oceans. *J. Phys. Oceanogr.*, **11 (11)**, 1443–1451.

Paulson, C. A. and J. J. Simpson, 1977 : Irradiance measurements in the upper ocean. *J. Phys. Oceanogr.*, **7 (6)**, 952–956.

Reverdin, G., P. Delecluse, C. Lévy, P. Andrich, A. Morlière, and J. M. Verstraete, 1991 : The near surface tropical atlantic in 1982-1984 : results from a numerical simulation and a data analysis. *Prog. Oceangr.*, **27**, 273–340.

Richtmyer, R. D. and K. W. Morton, 1967 : *Difference methods for initial-value problems*. Interscience Publisher, Second Edition, 405pp.

Robert, A. J., 1966 : The integration of a low order spectral form of the primitive meteorological equations. *J. Meteo. Soc. Japan*, **44, 2**.

Roullet, G. and G. Madec, 2000 : salt conservation, free surface, and varying levels : a new formulation for ocean general circulation models. *J. Geophys. Res*, **105**, 23,927–23,942.

Sadourny, R., 1975 : The dynamics of finite-difference models of the shallow-water equations. *J. Atmos. Sc.*, **32 (4)**, 680–689.

Sarmiento, J. L. and K. Bryan, 1982 : Ocean transport model for the north atlantic. *J. Geophys. Res*, **87**, 394–409.

Shchepetkin, A. F. and J. C. McWilliams, 2003 : A method for computing horizontal pressure-gradient force in an oceanic model with a nonaligned vertical coordinate. *J. Geophys. Res*, **108(C3)**, 3090, doi :10.1029/2001JC001 047.

———, 2005 : The regional oceanic modeling system (roms) - a split-explicit, free-surface, topography-following-coordinate oceanic modelr. *Ocean Modelling*, **9, 4**, 347–404.

Shchepetkin, A. F. and J. J. O'Brien, 1996 : A physically consistent formulation of lateral friction in shallow-water equation ocean models. *Mon. Wea. Rev.*, **124 (6)**, 1285–1300.

Song, Y. and D. Haidvogel, 1994 : A semi-implicit ocean circulation model using a generalized topography-following coordinate system authors :. *J. Comput. Phys.*, **115, 1**.

Song, Y. T., 1998 : A general pressure gradient formulation for ocean models. part i : Scheme design and diagnostic analysis. *Mon. Wea. Rev.*, **126 (12)**, 3213–3230.

Speich, S., 1992 : Etude du forçage de la circulation générale océanique par les détroits - cas de la mer d'alboran. Ph.D. thesis, Université Pierre et Marie Curie, Paris, France.

Speich, S., G. Madec, and M. Crépon, 1996 : The circulation in the alboran sea - a sensitivity study. *J. Phys. Oceanogr.*, **26**.

Steele, M., R. Morley, and W. Ermold, 2001 : Phc- a global ocean hydrography with a high-quality arctic ocean. *Journal of Climate*, **14 (9)**, 2079–2087.

Stein, C. A. and S. Stein, 1992 : A model for the global variation in oceanic depth and heat flow with lithospheric age. *Nature*, **359**, 123–129.

Thiem, O. and J. Berntsen, 2006 : Internal pressure errors in sigma-coordinate ocean models due to anisotropy. *Ocean Modelling*, **12, 1-2**.

Tréguier, A.-M., J. Dukowicz, and K. Bryan, 1996 : Properties of nonuniform grids used in ocean general circulation models. *J. Geophys. Res*, **101**, 20 877–20 881.

Tréguier, A. M., I. M. Held, and V. D. Larichev, 1997 : Parameterization of quasigeostrophic eddies in primitive equation ocean models. *J. Phys. Oceanogr.*, **27 (4)**, 567–580.

UNESCO, 1983 : *Algorithms for computation of fundamental property of sea water*. Techn. Paper in Mar. Sci, 44, UNESCO.

Valcke, S., 2006 : Oasis3 user guide (prism_2-5). Tech. rep., PRISM Support Initiative Report No 3, CERFACS, Toulouse, France, 64 pp.

Weatherly, G. L., 1984 : An estimate of bottom frictional dissipation by gulf stream fluctuations. *J. Mar. Res.*, **42, 2**, 289–301.

Weaver, A. J. and M. Eby, 1997 : On the numerical implementation of advection schemes for use in conjuction with various mixing parameterizations in the gfdl ocean model. *J. Phys. Oceanogr.*, **27**.

Webb, D. J., B. A. de Cuevas, and C. S. Richmond, 1998 : Improved advection schemes for ocean models. *J. Atmos. Ocean Tech.*, **15 (5)**, 1171–1187.

Willebrand, J., B. Barnier, C. Boning, C. Dieterich, P. D. Killworth, C. L. Provost, Y. Jia, J.-M. Molines, and A. L. New, 2001 : Circulation characteristics in three eddy-permitting models of the north atlantic. *Progress in Oceanography*, **48, 2**, 123–161.

Zalesak, S. T., 1979 : Fully multidimensional flux corrected transport algorithms for fluids. *J. Comput. Phys.*, **31**.

Zhang, R.-H. and M. Endoh, 1992 : A free surface general circulation model for the tropical pacific ocean. *J. Geophys. Res*, **97**, 11 237–11 255.