NEMO ocean engine

Gurvan Madec, and the NEMO team

gurvan.madec@locean-ipsl.umpc.fr

nemo_st@hermes.locean-ipsl.umpc.fr

January 2011

– version 3.3 –

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

ISSN No 1288-1619.

Table des matières

Abstract / Résumé

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

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

Disclaimer

Like all components of NEMO, the ocean component is developed under the CECILL license, which is a French adaptation of the GNU GPL (General Public License). Anyone may use it 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 NEMO.

NEMO reference in papers and other publications is as follows :

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

Additional information can be found on nemo-ocean.eu website.

1 Introduction

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

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

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

[1] OPA = Océan PArallélisé
[2] LIM= Louvain)la-neuve Ice Model
[3] TOP = Tracer in the Ocean Paradigm
[4] Both LOBSTER and PISCES are not acronyms just name

second (MKS) international system are used throughout.

The following chapters deal with the discrete equations. Chapter 3 presents the time domain. The model time stepping environment is a three level scheme in which the tendency terms of the equations are evaluated either centered in time, or forward, or backward depending of the nature of the term. Chapter 4 presents the space domain. The model is discretised on a staggered grid (Arakawa C grid) with masking of land areas. Vertical discretisation used depends on both how the bottom topography is represented and whether the free surface is linear or not. Full step or partial step z-coordinate or s-(terrain-following) coordinate is used with linear free surface (level position are then fixed in time). In non-linear free surface, the corresponding rescaled height coordinate formulation (z* or s*) is used (the level position then vary in time as a function of the sea surface heigh). The following two chapters (5 and 6) describe the discretisation of the prognostic equations for the active tracers and the momentum. Explicit, split-explicit and filtered free surface formulations are implemented. A number of numerical schemes are available for momentum advection, for the computation of the pressure gradients, as well as for the advection of tracers (second or higher order advection schemes, including positive ones).

Surface boundary conditions (chapter 7) 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 [**?**]. Two-way nesting is also available through an interface to the AGRIF package (Adaptative Grid Refinement in FORTRAN) [**?**].

Other model characteristics are the lateral boundary conditions (chapter 8). Global configurations of the model make use of the ORCA tripolar grid, with special north fold boundary condition. Free-slip or no-slip boundary conditions are allowed at land boundaries. Closed basin geometries as well as periodic domains and open boundary conditions are possible.

Physical parameterisations are described in chapters 9 and 10. The model includes an implicit treatment of vertical viscosity and diffusivity. The lateral Laplacian and biharmonic viscosity and diffusion can be rotated following a geopotential or neutral direction. There is an optional eddy induced velocity [**?**] with a space and time variable coefficient **?**. The model has vertical harmonic viscosity and diffusion with a space and time variable coefficient, with options to compute the coefficients with **?**, **?**, **?**, or **?** mixing schemes.

Model outputs management and specific online diagnostics are described in chapters **??**. The diagnostics includes the output of all the tendencies of the momentum and tracers equations, the output of tracers tendencies averaged over the time evolving mixed layer, the output of the tendencies of the barotropic vorticity equation, the computation of on-line floats trajectories... Chapter 11 describes a tool which reads in observation files (profile temperature and salinity, sea surface temperature, sea level anomaly and sea ice concentration) and calculates an interpolated model equivalent value at the observation location and nearest model timestep. Originally developed of data assimilation, it is a fan-

TAB. 1.1 – Organization of Chapters mimicking the one of the model directories.

Chapter 3	-	model time STePping environment
Chapter 4	DOM	model DOMain
Chapter 5	TRA	TRAcer equations (potential temperature and salinity)
Chapter 6	DYN	DYNamic equations (momentum)
Chapter 7	SBC	Surface Boundary Conditions
Chapter 8	LBC	Lateral Boundary Conditions (also OBC and BDY)
Chapter 9	LDF	Lateral DiFfusion (parameterisations)
Chapter 10	ZDF	vertical (Z) DiFfusion (parameterisations)
Chapter **??**	DIA	I/O and DIAgnostics (also IOM, FLO and TRD)
Chapter 11	OBS	OBServation and model comparison
Chapter 12	ASM	ASsiMilation increment
Chapter 13	SOL	Miscellaneous topics (including solvers)
Chapter 14	-	predefined configurations (including C1D)

tastic tool for model and data comparison. Chapter 12 describes how increments produced by data assimilation may be applied to the model equations. Finally, Chapter 14 provides a brief introduction to the pre-defined model configurations (water column model, ORCA and GYRE families of configurations).

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 briefly presented in Appendix D and a more complete document is available on the *NEMO* web site.

The model is organized with a high internal modularity based on physics. For example, each trend ($i.e.$, a term in the RHS of the prognostic equation) for momentum and tracers is computed in a dedicated module. To make it easier for the user to find his way around the code, the module names follow a three-letter rule. For example, *traldf.F90* is a module related to the TRAcers equation, computing the Lateral DiFfussion. Furthermore, modules are organized in a few directories that correspond to their category, as indicated by the first three letters of their name (Tab. 1.1).

The manual mirrors the organization of the model. After the presentation of the continuous equations (Chapter 2), the following chapters refer to specific terms of the equations each associated with a group of modules (Tab. 1.1).

Changes between releases

NEMO/OPA, like all research tools, is in perpetual evolution. The present document describes the OPA version include in the release 3.3 of NEMO. This release differs significantly from version 8, documented in **?**.

• The main modifications from OPA v8 and NEMO/OPA v3.2 are :

1. transition to full native FORTRAN 90, deep code restructuring and drastic reduction of CPP keys ;
2. introduction of partial step representation of bottom topography [**???**] ;
3. partial reactivation of a terrain-following vertical coordinate (s- and hybrid s-z) with the addition of several options for pressure gradient computation [5] ;
4. more choices for the treatment of the free surface : full explicit, split-explicit or filtered schemes, and suppression of the rigid-lid option ;
5. non linear free surface associated with the rescaled height coordinate z* or s ;
6. additional schemes for vector and flux forms of the momentum advection ;
7. additional advection schemes for tracers ;
8. implementation of the AGRIF package (Adaptative Grid Refinement in FORTRAN) [**?**] ;
9. online diagnostics : tracers trend in the mixed layer and vorticity balance ;
10. rewriting of the I/O management with the use of an I/O server ;
11. generalized ocean-ice-atmosphere-CO2 coupling interface, interfaced with OASIS 3 coupler ;
12. surface module (SBC) that simplify the way the ocean is forced and include two bulk formulea (CLIO and CORE) and which includes an on-the-fly interpolation of input forcing fields ;
13. RGB light penetration and optional use of ocean color
14. major changes in the TKE schemes : it now includes a Langmuir cell parameterization [**?**], the **?** surface wave breaking parameterization, and has a time discretization which is energetically consistent with the ocean model equations [**??**] ;
15. tidal mixing parametrisation (bottom intensification) + Indonesian specific tidal mixing [**?**] ;
16. introduction of LIM-3, the new Louvain-la-Neuve sea-ice model (C-grid rheology and new thermodynamics including bulk ice salinity) [**??**]

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

• The main modifications from NEMO/OPA v3.2 and v3.3 are :

1. introduction of a modified leapfrog-Asselin filter time stepping scheme [**?**] ;
2. additional scheme for iso-neutral mixing [**?**], although it is still a "work in progress" ;
3. a rewriting of the bottom boundary layer scheme, following **?** ;
4. addition of a Generic Length Scale vertical mixing scheme, following **?** ;
5. addition of the atmospheric pressure as an external forcing on both ocean and sea-ice dynamics ;
6. addition of a diurnal cycle on solar radiation [**?**] ;
7. river runoffs added through a non-zero depth, and having its own temperature and salinity ;
8. CORE II normal year forcing set as the default forcing of ORCA2-LIM configuration ;
9. generalisation of the use of *fldread.F90* for all input fields (ocean climatology, sea-ice damping...) ;
10. addition of an on-line observation and model comparison (thanks to NEMOVAR project) ;
11. optional application of an assimilation increment (thanks to NEMOVAR project) ;
12. coupling interface adjusted for WRF atmospheric model ;
13. C-grid ice rheology now available fro both LIM-2 and LIM-3 [**?**] ;
14. LIM-3 ice-ocean momentum coupling applied to LIM-2 ;
15. a deep re-writting and simplification of the off-line tracer component (OFF_SRC) ;
16. the merge of passive and active advection and diffusion modules ;
17. Use of the Flexible Configuration Manager (FCM) to build configurations, generate the Makefile and produce the executable ;
18. Linear-tangent and Adjoint component (TAM) added, phased with v3.0

In addition, several minor modifications in the coding have been introduced with the constant concern of improving the model performance.

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 the buoyancy force (this removes convective processes from the initial Navier-Stokes equations and so convective processes must be parameterized instead)

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

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

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

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

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

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

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

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

where ∇ is the generalised derivative vector operator in $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ directions, t is the time, z is the vertical coordinate, ρ is the *in situ* density given by the equation of state (2.1f), ρ_o is a reference density, p the pressure, $f = 2\mathbf{\Omega} \cdot \mathbf{k}$ is the Coriolis acceleration (where $\mathbf{\Omega}$ is the Earth's angular velocity vector), and g is the gravitational acceleration. $\mathbf{D}^{\mathbf{U}}$, D^T and D^S are the parameterisations of small-scale physics for momentum, temperature and salinity, and $\mathbf{F}^{\mathbf{U}}$, F^T and F^S surface forcing terms. Their nature and formulation are discussed in §2.5 and page §2.1.2.

.

2.1.2 Boundary Conditions

An ocean is bounded by complex coastlines, bottom topography at its base and an air-sea or ice-sea interface at its top. These boundaries can be defined by two surfaces, $z = -H(i, j)$ and $z = \eta(i, j, k, t)$, where H is the depth of the ocean bottom and η is the height of the sea surface. Both H and η are usually referenced to a given surface, $z = 0$, chosen as a mean sea surface (Fig. 2.1). Through these two boundaries, the ocean can exchange fluxes of heat, fresh water, salt, and momentum with the solid earth, the continental margins, the sea ice and the atmosphere. However, some of these fluxes are so weak that even on climatic time scales of thousands of years they can be neglected. In the following, we briefly review the fluxes exchanged at the interfaces between the ocean and the other components of the earth system.

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

Solid earth - ocean interface : heat and salt fluxes through the sea floor are small, except in special areas of little extent. They are usually neglected in the model [1]. The

[1] In fact, it has been shown that the heat flux associated with the solid Earth cooling (*i.e.*the geothermal heating) is not negligible for the thermohaline circulation of the world ocean (see 5.4.3).

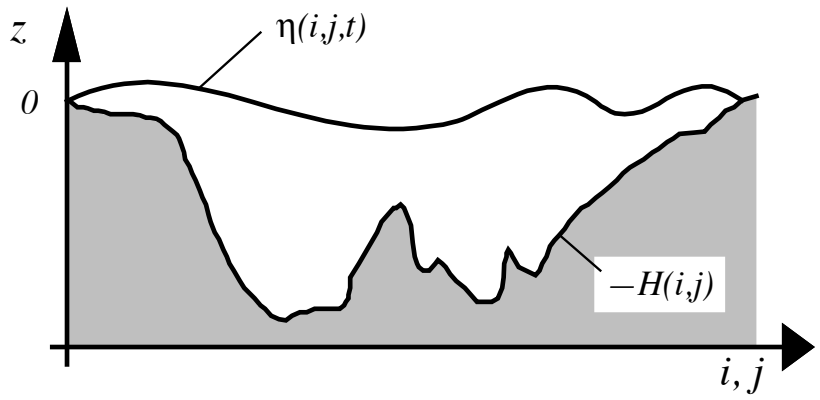

FIG. 2.1 – The ocean is bounded by two surfaces, $z = -H(i,j)$ and $z = \eta(i,j,t)$, where H is the depth of the sea floor and η the height of the sea surface. Both H and η are referenced to $z = 0$.

boundary condition is thus set to no flux of heat and salt across solid boundaries. For momentum, the situation is different. There is no flow across solid boundaries, $i.e.$ the velocity normal to the ocean bottom and coastlines is zero (in other words, the bottom velocity is parallel to solid boundaries). This kinematic boundary condition can be expressed as :

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

In addition, the ocean exchanges momentum with the earth through frictional processes. Such momentum transfer occurs at small scales in a boundary layer. It must be parameterized in terms of turbulent fluxes using bottom and/or lateral boundary conditions. Its specification depends on the nature of the physical parameterisation used for $\mathbf{D}^{\mathbf{U}}$ in (2.1a). It is discussed in §2.5.1, page 31.

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

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

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

Sea ice - ocean interface : the ocean and sea ice exchange heat, salt, fresh water and momentum. The sea surface temperature is constrained to be at the freezing point

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

2.2 The Horizontal Pressure Gradient

2.2.1 Pressure Formulation

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

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

Two strategies can be considered for the surface pressure term : (a) introduce of a new variable η, the free-surface elevation, for which a prognostic equation can be established and solved ; (b) assume that the ocean surface is a rigid lid, on which the pressure (or its horizontal gradient) can be diagnosed. When the former strategy is used, one solution of the free-surface elevation consists of the excitation of external gravity waves. The flow is barotropic and the surface moves up and down with gravity as the restoring force. The phase speed of such waves is high (some hundreds of metres per second) so that the time step would have to be very short if they were present in the model. The latter strategy filters out these waves since the rigid lid approximation implies $\eta = 0$, $i.e.$ the sea surface is the surface $z = 0$. This well known approximation increases the surface wave speed to infinity and modifies certain other longwave dynamics ($e.g.$ barotropic Rossby or planetary waves). The rigid-lid hypothesis is an obsolescent feature in modern OGCMs. It has been available until the release 3.1 of *NEMO* , and it has been removed in release 3.2 and followings. Only the free surface formulation is now described in the this document (see the next sub-section).

2.2.2 Free Surface Formulation

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

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

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

Allowing the air-sea interface to move introduces the external gravity waves (EGWs) as a class of solution of the primitive equations. These waves are barotropic because of

hydrostatic assumption, and their phase speed is quite high. Their time scale is short with respect to the other processes described by the primitive equations.

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

• If one is interested in EGWs, in particular the tides and their interaction with the baroclinic structure of the ocean (internal waves) possibly in shallow seas, then a non linear free surface is the most appropriate. This means that no approximation is made in (2.5) and that the variation of the ocean volume is fully taken into account. Note that in order to study the fast time scales associated with EGWs it is necessary to minimize time filtering effects (use an explicit time scheme with very small time step, or a split-explicit scheme with reasonably small time step, see §6.5.1 or §6.5.2.

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

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

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

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

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

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

the nonlinear free surface equation, the time evolution of the total salt content is

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

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

$$\frac{\partial}{\partial t}\int\limits_{D} S\ dv = -\int\limits_{S} S\ \frac{\partial \eta}{\partial t} ds \tag{2.8}$$

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

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

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

2.3 Curvilinear *z*-coordinate System

2.3.1 Tensorial Formalism

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

Let (*i*,*j*,*k*) be a set of orthogonal curvilinear coordinates on the sphere associated with the positively oriented orthogonal set of unit vectors (**i**,**j**,**k**) linked to the earth such that **k** is the local upward vector and (**i**,**j**) are two vectors orthogonal to **k**, $i.e.$ along geopotential surfaces (Fig.2.2). Let (λ, φ, z) be the geographical coordinate system in which a position is defined by the latitude $\varphi(i,j)$, the longitude $\lambda(i,j)$ and the distance from the centre of the earth $a + z(k)$ where a is the earth's radius and z the altitude above a reference sea level (Fig.2.2). The local deformation of the curvilinear coordinate system is given by e_1, e_2 and e_3, the three scale factors :

$$
\begin{aligned}
e_1 &= (a+z) \left[\left(\frac{\partial \lambda}{\partial i} \cos\varphi \right)^2 + \left(\frac{\partial \varphi}{\partial i} \right)^2 \right]^{1/2} \\
e_2 &= (a+z) \left[\left(\frac{\partial \lambda}{\partial j} \cos\varphi \right)^2 + \left(\frac{\partial \varphi}{\partial j} \right)^2 \right]^{1/2} \\
e_3 &= \left(\frac{\partial z}{\partial k} \right)
\end{aligned}
\tag{2.10}
$$

Since the ocean depth is far smaller than the earth's radius, $a + z$, can be replaced by a in (2.10) (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

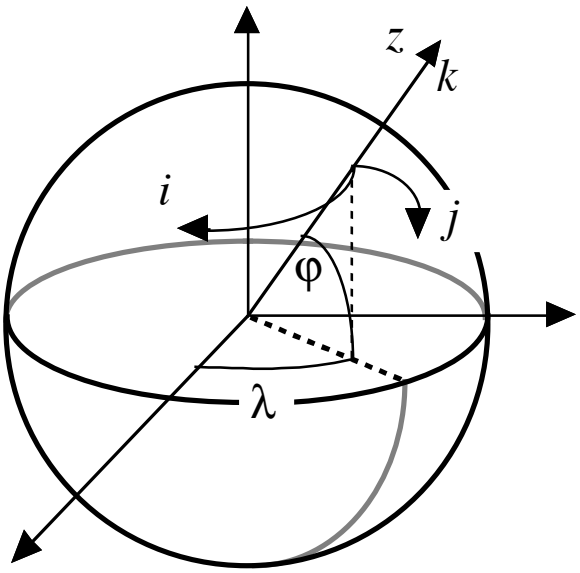

FIG. 2.2 – the geographical coordinate system (λ, φ, z) and the curvilinear coordinate system (**i**,**j**,**k**).

(2.1f)) can be written in the tensorial form, invariant in any orthogonal horizontal curvilinear coordinate system transformation :

$$\nabla q = \frac{1}{e_1}\frac{\partial q}{\partial i}\,\mathbf{i} + \frac{1}{e_2}\frac{\partial q}{\partial j}\,\mathbf{j} + \frac{1}{e_3}\frac{\partial q}{\partial k}\,\mathbf{k} \tag{2.11a}$$

$$\nabla \cdot \mathbf{A} = \frac{1}{e_1\, e_2}\left[\frac{\partial\,(e_2\, a_1)}{\partial i} + \frac{\partial\,(e_1\, a_2)}{\partial j}\right] + \frac{1}{e_3}\left[\frac{\partial a_3}{\partial k}\right] \tag{2.11b}$$

$$\begin{aligned}\nabla \times \mathbf{A} = \left[\frac{1}{\mathrm{e_2}}\frac{\partial \mathrm{a_3}}{\partial \mathrm{j}} - \frac{1}{\mathrm{e_3}}\frac{\partial \mathrm{a_2}}{\partial \mathrm{k}}\right]\,\mathbf{i} + \left[\frac{1}{e_3}\frac{\partial a_1}{\partial k} - \frac{1}{e_1}\frac{\partial a_3}{\partial i}\right]\,\mathbf{j} \\ + \frac{1}{e_1 e_2}\left[\frac{\partial\,(e_2 a_2)}{\partial i} - \frac{\partial\,(e_1 a_1)}{\partial j}\right]\,\mathbf{k}\end{aligned} \tag{2.11c}$$

$$\Delta q = \nabla \cdot (\nabla q) \tag{2.11d}$$

$$\Delta \mathbf{A} = \nabla\,(\nabla \cdot \mathbf{A}) - \nabla \times (\nabla \times \mathbf{A}) \tag{2.11e}$$

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 (2.11a)) to (2.11e). 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 :

$$\zeta = \frac{1}{e_1 e_2}\left[\frac{\partial\left(e_2\, v\right)}{\partial i} - \frac{\partial\left(e_1\, u\right)}{\partial j}\right] \tag{2.12}$$

$$\chi = \frac{1}{e_1 e_2}\left[\frac{\partial\left(e_2\, u\right)}{\partial i} + \frac{\partial\left(e_1\, v\right)}{\partial j}\right] \tag{2.13}$$

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

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

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

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

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

$$\left[(\nabla \times \mathbf{U}) \times \mathbf{U} + \frac{1}{2}\nabla\left(\mathbf{U}^2\right)\right]_h = \zeta\ \mathbf{k} \times \mathbf{U}_h + \frac{1}{2}\nabla_h\left(\mathbf{U}_h^2\right) + \frac{1}{e_3} w \frac{\partial \mathbf{U}_h}{\partial k} \tag{2.14}$$

This is the so-called *vector invariant form* of the momentum advection term. For some purposes, it can be advantageous to write this term in the so-called flux form, *i.e.* to write it as the divergence of fluxes. For example, the first component of (2.14) (the i-component) is transformed as follows :

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

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

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

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

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

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

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

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

$$\left[(\nabla\times\mathbf{U})\times\mathbf{U} + \frac{1}{2}\nabla\left(\mathbf{U}^2\right)\right]_h = \nabla\cdot\begin{pmatrix}\mathbf{U}\,u\\ \mathbf{U}\,v\end{pmatrix} + \frac{1}{e_1 e_2}\left(v\frac{\partial e_2}{\partial i} - u\frac{\partial e_1}{\partial j}\right)\mathbf{k}\times\mathbf{U}_h \quad (2.15)$$

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

$$f \to f + \frac{1}{e_1\, e_2}\left(v\frac{\partial e_2}{\partial i} - u\frac{\partial e_1}{\partial j}\right) \quad (2.16)$$

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

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

• **Vector invariant form of the momentum equations** :

$$\frac{\partial u}{\partial t} = +(\zeta+f)\;v - \frac{1}{2\,e_1}\frac{\partial}{\partial i}\left(u^2+v^2\right) - \frac{1}{e_3}w\frac{\partial u}{\partial k} - \frac{1}{e_1}\frac{\partial}{\partial i}\left(\frac{p_s+p_h}{\rho_o}\right) + D_u^{\mathbf{U}} + F_u^{\mathbf{U}} \quad (2.17a)$$

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

• **flux form of the momentum equations** :

$$\frac{\partial u}{\partial t} = + \left(f + \frac{1}{e_1\, e_2} \left(v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right) \right) v - \frac{1}{e_1\, e_2} \left(\frac{\partial (e_2\, u\, u)}{\partial i} + \frac{\partial (e_1\, v\, u)}{\partial j} \right) - \frac{1}{e_3} \frac{\partial (w\, u)}{\partial k} - \frac{1}{e_1} \frac{\partial}{\partial i} \left(\frac{p_s + p_h}{\rho_o} \right) + D_u^{\mathbf{U}} + F_u^{\mathbf{U}} \quad (2.18a)$$

$$\frac{\partial v}{\partial t} = - \left(f + \frac{1}{e_1\, e_2} \left(v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right) \right) u - \frac{1}{e_1\, e_2} \left(\frac{\partial (e_2\, u\, v)}{\partial i} + \frac{\partial (e_1\, v\, v)}{\partial j} \right) - \frac{1}{e_3} \frac{\partial (w\, v)}{\partial k} - \frac{1}{e_2} \frac{\partial}{\partial j} \left(\frac{p_s + p_h}{\rho_o} \right) + D_v^{\mathbf{U}} + F_v^{\mathbf{U}} \quad (2.18b)$$

where ζ, the relative vorticity, is given by (2.12) and p_s, the surface pressure, is given by :

$$p_s = \begin{cases} \rho\, g\, \eta & \text{standard free surface} \\ \rho\, g\, \eta + \rho_o\, \mu\, \dfrac{\partial \eta}{\partial t} & \text{filtered free surface} \end{cases} \quad (2.19)$$

with η is solution of (2.5)

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

$$\frac{\partial w}{\partial k} = -\chi\, e_3 \quad (2.20)$$

$$\frac{\partial p_h}{\partial k} = -\rho\, g\, e_3 \quad (2.21)$$

where the divergence of the horizontal velocity, χ is given by (2.13).

• *tracer equations* :

$$\frac{\partial T}{\partial t} = -\frac{1}{e_1 e_2} \left[\frac{\partial (e_2 T\, u)}{\partial i} + \frac{\partial (e_1 T\, v)}{\partial j} \right] - \frac{1}{e_3} \frac{\partial (T\, w)}{\partial k} + D^T + F^T \quad (2.22)$$

$$\frac{\partial S}{\partial t} = -\frac{1}{e_1 e_2} \left[\frac{\partial (e_2 S\, u)}{\partial i} + \frac{\partial (e_1 S\, v)}{\partial j} \right] - \frac{1}{e_3} \frac{\partial (S\, w)}{\partial k} + D^S + F^S \quad (2.23)$$

$$\rho = \rho\,(T, S, z(k)) \quad (2.24)$$

The expression of $\mathbf{D}^U$, D^S and D^T depends on the subgrid scale parameterisation used. It will be defined in §2.5.1. The nature and formulation of $\mathbf{F}^{\mathbf{U}}$, F^T and F^S, the surface forcing terms, are discussed in Chapter 7.

2.4 Curvilinear generalised vertical coordinate System

The ocean domain presents a huge diversity of situation in the vertical. First the ocean surface is a time dependent surface (moving surface). Second the ocean floor depends on the geographical position, varying from more than 6,000 meters in abyssal trenches to zero at the coast. Last but not least, the ocean stratification exerts a strong barrier to vertical motions and mixing. Therefore, in order to represent the ocean with respect to the first point a space and time dependent vertical coordinate that follows the variation of the sea surface height $e.g.$ an z*-coordinate ; for the second point, a space variation to fit the change of bottom topography $e.g.$ a terrain-following or σ-coordinate ; and for the third point, one will be tempted to use a space and time dependent coordinate that follows the isopycnal surfaces, $e.g.$ an isopycnic coordinate.

In order to satisfy two or more constrains one can even be tempted to mixed these coordinate systems, as in HYCOM (mixture of z-coordinate at the surface, isopycnic coordinate in the ocean interior and σ at the ocean bottom) [**?**] or OPA (mixture of z-coordinate in vicinity the surface and steep topography areas and σ-coordinate elsewhere) [**?**] among others.

In fact one is totally free to choose any space and time vertical coordinate by introducing an arbitrary vertical coordinate :

$$s = s(i, j, k, t) \tag{2.25}$$

with the restriction that the above equation gives a single-valued monotonic relationship between s and k, when i, j and t are held fixed. (2.25) is a transformation from the (i, j, k, t) coordinate system with independent variables into the (i, j, s, t) generalised coordinate system with s depending on the other three variables through (2.25). This so-called *generalised vertical coordinate* [**?**] is in fact an Arbitrary Lagrangian–Eulerian (ALE) coordinate. Indeed, choosing an expression for s is an arbitrary choice that determines which part of the vertical velocity (defined from a fixed referential) will cross the levels (Eulerian part) and which part will be used to move them (Lagrangian part). The coordinate is also sometime referenced as an adaptive coordinate [**?**], since the coordinate system is adapted in the course of the simulation. Its most often used implementation is via an ALE algorithm, in which a pure lagrangian step is followed by regridding and remapping steps, the later step implicitly embedding the vertical advection [**???**]. Here we follow the [**?**] strategy : a regridding step (an update of the vertical coordinate) followed by an eulerian step with an explicit computation of vertical advection relative to the moving s-surfaces.

the generalized vertical coordinates used in ocean modelling are not orthogonal, which contrasts with many other applications in mathematical physics. Hence, it is useful to keep in mind the following properties that may seem odd on initial encounter.

The horizontal velocity in ocean models measures motions in the horizontal plane, perpendicular to the local gravitational field. That is, horizontal velocity is mathematically the same regardless the vertical coordinate, be it geopotential, isopycnal, pressure, or terrain following. The key motivation for maintaining the same horizontal velocity component is that the hydrostatic and geostrophic balances are dominant in the large-scale ocean. Use of an alternative quasi-horizontal velocity, for example one oriented parallel to the generalized surface, would lead to unacceptable numerical errors. Correspondingly, the vertical direction is anti-parallel to the gravitational force in all of the coordinate systems. We do not choose the alternative of a quasi-vertical direction oriented normal to the surface of a constant generalized vertical coordinate.

It is the method used to measure transport across the generalized vertical coordinate surfaces which differs between the vertical coordinate choices. That is, computation of the dia-surface velocity component represents the fundamental distinction between the various coordinates. In some models, such as geopotential, pressure, and terrain following, this transport is typically diagnosed from volume or mass conservation. In other models, such as isopycnal layered models, this transport is prescribed based on assumptions about the physical processes producing a flux across the layer interfaces.

In this section we first establish the PE in the generalised vertical s-coordinate, then we discuss the particular cases available in *NEMO* , namely z, z*, s, and $\tilde{z}$.

2.4.1 The *s*-coordinate Formulation

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

$$\sigma_1 = \frac{1}{e_1} \left.\frac{\partial z}{\partial i}\right|_s \quad \text{, and} \quad \sigma_2 = \frac{1}{e_2} \left.\frac{\partial z}{\partial j}\right|_s \tag{2.26}$$

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

$$\omega = w - e_3 \frac{\partial s}{\partial t} - \sigma_1 u - \sigma_2 v \tag{2.27}$$

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

* momentum equation :

$$\frac{1}{e_3}\frac{\partial (e_3\, u)}{\partial t} = +(\zeta + f)\, v - \frac{1}{2\, e_1}\frac{\partial}{\partial i}\left(u^2 + v^2\right) - \frac{1}{e_3}\omega\frac{\partial u}{\partial k} - \frac{1}{e_1}\frac{\partial}{\partial i}\left(\frac{p_s + p_h}{\rho_o}\right) + g\frac{\rho}{\rho_o}\sigma_1 + D_u^{\mathbf{U}} + F_u^{\mathbf{U}} \tag{2.28}$$

$$\frac{1}{e_3}\frac{\partial (e_3\, v)}{\partial t} = -(\zeta + f)\, u - \frac{1}{2\, e_2}\frac{\partial}{\partial j}\left(u^2 + v^2\right) - \frac{1}{e_3}\omega\frac{\partial v}{\partial k} - \frac{1}{e_2}\frac{\partial}{\partial j}\left(\frac{p_s + p_h}{\rho_o}\right) + g\frac{\rho}{\rho_o}\sigma_2 + D_v^{\mathbf{U}} + F_v^{\mathbf{U}} \tag{2.29}$$

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

$$\frac{\partial e_3}{\partial t} + e_3\, \chi + \frac{\partial \omega}{\partial s} = 0 \qquad \text{with} \quad \chi = \frac{1}{e_1 e_2 e_3}\left[\frac{\partial (e_2 e_3\, u)}{\partial i} + \frac{\partial (e_1 e_3\, v)}{\partial j}\right] \tag{2.30}$$

* tracer equations :

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

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

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

2.4.2 Curvilinear z*–coordinate System

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

The z* coordinate approach is an unapproximated, non-linear free surface implementation which allows one to deal with large amplitude free-surface variations relative to the

vertical resolution [**?**]. In the z* formulation, the variation of the column thickness due to sea-surface undulations is not concentrated in the surface level, as in the z-coordinate formulation, but is equally distributed over the full water column. Thus vertical levels naturally follow sea-surface variations, with a linear attenuation with depth, as illustrated by figure fig.1c . Note that with a flat bottom, such as in fig.1c, the bottom-following z coordinate and z* are equivalent. The definition and modified oceanic equations for the rescaled vertical coordinate z*, including the treatment of fresh-water flux at the surface, are detailed in Adcroft and Campin (2004). The major points are summarized here. The position (z*) and vertical discretization (z*) are expressed as :

$$H + z* = (H + z)/r \quad \text{and } \delta z* = \delta z/r \quad \text{with } r = \frac{H + \eta}{H} \tag{2.33}$$

Since the vertical displacement of the free surface is incorporated in the vertical coordinate z*, the upper and lower boundaries are at fixed z* position, $z* = 0$ and $z* = -H$ respectively. Also the divergence of the flow field is no longer zero as shown by the conti-

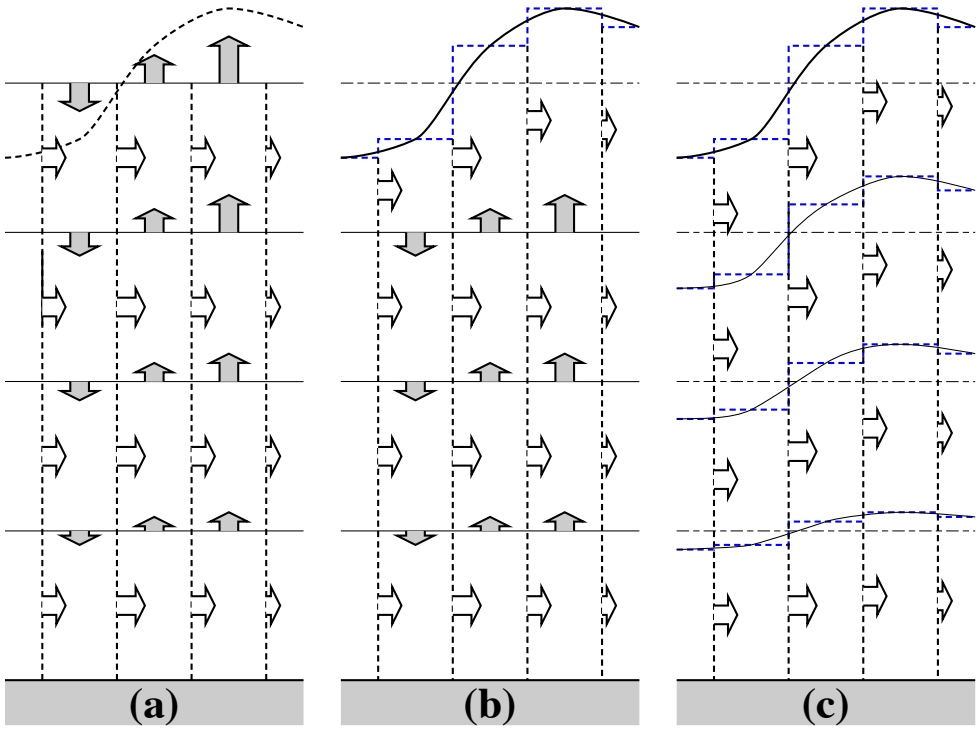

FIG. 2.3 – (a) z-coordinate in linear free-surface case ; (b) $z-$coordinate in non-linear free surface case ; (c) re-scaled height coordinate (become popular as the z*-coordinate [**?**]).

nuity equation :

$$\frac{\partial r}{\partial t} = \nabla_{z^*} \cdot (r\ \mathbf{U}_{\mathrm{h}})\,(r\ w^*) = 0$$

To overcome problems with vanishing surface and/or bottom cells, we consider the zstar coordinate

$$z^\star = H\left(\frac{z-\eta}{H+\eta}\right) \tag{2.34}$$

This coordinate is closely related to the "eta" coordinate used in many atmospheric models (see Black (1994) for a review of eta coordinate atmospheric models). It was originally used in ocean models by Stacey et al. (1995) for studies of tides next to shelves, and it has been recently promoted by Adcroft and Campin (2004) for global climate modelling.

The surfaces of constant $z^\star$ are quasi-horizontal. Indeed, the $z^\star$ coordinate reduces to z when η is zero. In general, when noting the large differences between undulations of the bottom topography versus undulations in the surface height, it is clear that surfaces constant $z^\star$ are very similar to the depth surfaces. These properties greatly reduce difficulties of computing the horizontal pressure gradient relative to terrain following sigma models discussed in §2.4.3. Additionally, since $z^\star$ when $\eta = 0$, no flow is spontaneously generated in an unforced ocean starting from rest, regardless the bottom topography. This behaviour is in contrast to the case with "s"-models, where pressure gradient errors in the presence of nontrivial topographic variations can generate nontrivial spontaneous flow from a resting state, depending on the sophistication of the pressure gradient solver. The quasi-horizontal nature of the coordinate surfaces also facilitates the implementation of neutral physics parameterizations in $z^\star$ models using the same techniques as in z-models (see Chapters 13-16 of **?**) for a discussion of neutral physics in z-models, as well as Section §9.2 in this document for treatment in *NEMO*).

The range over which $z^\star$ varies is time independent $-H \le z^\star \le 0$. Hence, all cells remain nonvanishing, so long as the surface height maintains $\eta > ?H$. This is a minor constraint relative to that encountered on the surface height when using $s = z$ or $s = z-\eta$.

Because $z^\star$ has a time independent range, all grid cells have static increments ds, and the sum of the ver tical increments yields the time independent ocean depth The $z^\star$ coordinate is therefore invisible to undulations of the free surface, since it moves along with the free surface. This proper ty means that no spurious ver tical transpor t is induced across surfaces of constant $z^\star$ by the motion of external gravity waves. Such spurious transpor t can be a problem in z-models, especially those with tidal forcing. Quite generally, the time independent range for the $z^\star$ coordinate is a very convenient proper ty that allows for a nearly arbitrary ver tical resolution even in the presence of large amplitude fluctuations of the surface height, again so long as $\eta > -H$.

2.4.3 Curvilinear Terrain-following *s*–coordinate

Introduction

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

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

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

The second term in (2.35) depends on the tilt of the coordinate surface and introduces a truncation error that is not present in a z-model. In the special case of a $\sigma-$coordinate (i.e. a depth-normalised coordinate system $\sigma = z/H$), **?** and **?** have given estimates of the magnitude of this truncation error. It depends on topographic slope, stratification, horizontal and vertical resolution, the equation of state, and the finite difference scheme. This error limits the possible topographic slopes that a model can handle at a given horizontal and vertical resolution. This is a severe restriction for large-scale applications using realistic bottom topography. The large-scale slopes require high horizontal resolution, and the computational cost becomes prohibitive. This problem can be at least partially overcome

by mixing s-coordinate and step-like representation of bottom topography [**???**]. However, the definition of the model domain vertical coordinate becomes then a non-trivial thing for a realistic bottom topography : a envelope topography is defined in s-coordinate on which a full or partial step bottom topography is then applied in order to adjust the model depth to the observed one (see §4.3.

For numerical reasons a minimum of diffusion is required along the coordinate surfaces of any finite difference model. It causes spurious diapycnal mixing when coordinate surfaces do not coincide with isoneutral surfaces. This is the case for a z-model as well as for a s-model. However, density varies more strongly on $s-$surfaces than on horizontal surfaces in regions of large topographic slopes, implying larger diapycnal diffusion in a s-model than in a z-model. Whereas such a diapycnal diffusion in a z-model tends to weaken horizontal density (pressure) gradients and thus the horizontal circulation, it usually reinforces these gradients in a s-model, creating spurious circulation. For example, imagine an isolated bump of topography in an ocean at rest with a horizontally uniform stratification. Spurious diffusion along s-surfaces will induce a bump of isoneutral surfaces over the topography, and thus will generate there a baroclinic eddy. In contrast, the ocean will stay at rest in a z-model. As for the truncation error, the problem can be reduced by introducing the terrain-following coordinate below the strongly stratified portion of the water column ($i.e.$ the main thermocline) [**?**]. An alternate solution consists of rotating the lateral diffusive tensor to geopotential or to isoneutral surfaces (see §2.5.2. Unfortunately, the slope of isoneutral surfaces relative to the s-surfaces can very large, strongly exceeding the stability limit of such a operator when it is discretized (see Chapter 9).

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

2.4.4 Curvilinear $\tilde{z}$–coordinate

The $\tilde{z}$-coordinate has been developed by **?**. It is not available in the current version of *NEMO* .

2.5 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 $\mathbf{D}^{v\mathbf{U}}$, D^{vS} and D^{vT}. The formulation of these terms and their underlying physics are briefly discussed in the next two subsections.

2.5.1 Vertical Subgrid Scale Physics

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

$$\begin{aligned} \mathbf{D}^{v\mathbf{U}} &= \frac{\partial}{\partial z}\left(A^{vm} \frac{\partial \mathbf{U}_h}{\partial z} \right) , \\ D^{vT} &= \frac{\partial}{\partial z}\left(A^{vT} \frac{\partial T}{\partial z} \right) , \quad D^{vS} = \frac{\partial}{\partial z}\left(A^{vT} \frac{\partial S}{\partial z} \right) \end{aligned} \tag{2.36}$$

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. 7 and 10 and §5.5). All the vertical physics is embedded in the specification of the eddy coefficients. They can be assumed to be either constant, or function of the local fluid properties ($e.g.$ Richardson number, Brunt-Vaisälä frequency...),

or computed from a turbulent closure model. The choices available in *NEMO* are discussed in §10).

2.5.2 Lateral Diffusive and Viscous Operators Formulation

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

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

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

All these parameterisations of subgrid scale physics present advantages and draw-

backs. There are not all available in *NEMO* . In the z-coordinate formulation, five options are offered for active tracers (temperature and salinity) : second order geopotential operator, second order isoneutral operator, **?** parameterisation, fourth order geopotential operator, and various slightly diffusive advection schemes. The same options are available for momentum, except **?** parameterisation which only involves tracers. In the s-coordinate formulation, additional options are offered for tracers : second order operator acting along $s-$surfaces, and for momentum : fourth order operator acting along $s-$surfaces (see §9).

lateral second order tracer diffusive operator

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

$$D^{lT} = \nabla.\left(A^{lT}\ \Re\ \nabla T\right) \qquad \text{with} \qquad \Re = \begin{pmatrix} 1 & 0 & -r_1 \\ 0 & 1 & -r_2 \\ -r_1 & -r_2 & r_1^2 + r_2^2 \end{pmatrix} \tag{2.37}$$

where r_1 and r_2 are the slopes between the surface along which the diffusive operator acts and the model level ($e.g.$ z- or s-surfaces). Note that the formulation (2.37) is exact for the rotation between geopotential and s-surfaces, while it is only an approximation for the rotation between isoneutral and z- or s-surfaces. Indeed, in the latter case, two assumptions are made to simplify (2.37) [**?**]. First, the horizontal contribution of the dianeutral mixing is neglected since the ratio between iso and dia-neutral diffusive coefficients is known to be several orders of magnitude smaller than unity. Second, the two isoneutral directions of diffusion are assumed to be independent since the slopes are generally less than 10^{-2} in the ocean (see Appendix B).

For *geopotential* diffusion, r_1 and r_2 are the slopes between the geopotential and computational surfaces : in z-coordinates they are zero ($r_1 = r_2 = 0$) while in s-coordinate (including z* case) they are equal to σ_1 and σ_2, respectively (see (2.26)).

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

$$r_1 = \frac{e_3}{e_1}\left(\frac{\partial \rho}{\partial i}\right)\left(\frac{\partial \rho}{\partial k}\right)^{-1}, \quad r_1 = \frac{e_3}{e_1}\left(\frac{\partial \rho}{\partial i}\right)\left(\frac{\partial \rho}{\partial k}\right)^{-1} \tag{2.38}$$

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

$$D^{lT} = \nabla \cdot \left(A^{lT}\ \Re\ \nabla T\right) + \nabla \cdot (\mathbf{U}^*\, T) \tag{2.39}$$

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

$$\begin{aligned} u^* &= +\frac{1}{e_3}\frac{\partial}{\partial k}\left[A^{eiv}\ \tilde{r}_1\right] \\ v^* &= +\frac{1}{e_3}\frac{\partial}{\partial k}\left[A^{eiv}\ \tilde{r}_2\right] \\ w^* &= -\frac{1}{e_1 e_2}\left[\frac{\partial}{\partial i}\left(A^{eiv}\ e_2\, \tilde{r}_1\right) + \frac{\partial}{\partial j}\left(A^{eiv}\ e_1\, \tilde{r}_2\right)\right] \end{aligned} \tag{2.40}$$

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

$$\tilde{r}_n = \begin{cases} r_n & \text{in } z\text{-coordinate} \\ r_n + \sigma_n & \text{in } z^* \text{ and } s\text{-coordinates} \end{cases} \qquad \text{where } n = 1, 2 \tag{2.41}$$

The normal component of the eddy induced velocity is zero at all the boundaries. This can be achieved in a model by tapering either the eddy coefficient or the slopes to zero in the vicinity of the boundaries. The latter strategy is used in *NEMO* (cf. Chap. 9).

lateral fourth order tracer diffusive operator

The lateral fourth order tracer diffusive operator is defined by :

$$D^{lT} = \Delta \left(A^{lT} \, \Delta T \right) \qquad \text{where } D^{lT} = \Delta \left(A^{lT} \, \Delta T \right) \tag{2.42}$$

It is the second order operator given by (2.37) 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 (2.11e) to the horizontal velocity vector (see Appendix B) :

$$\begin{aligned} \mathbf{D}^{l\mathbf{U}} &= \nabla_h \left(A^{lm} \chi \right) - \nabla_h \times \left(A^{lm} \, \zeta \, \mathbf{k} \right) \\ &= \begin{pmatrix} \dfrac{1}{e_1} \dfrac{\partial \left(A^{lm} \chi \right)}{\partial i} - \dfrac{1}{e_2 e_3} \dfrac{\partial \left(A^{lm} \, e_3 \zeta \right)}{\partial j} \\ \dfrac{1}{e_2} \dfrac{\partial \left(A^{lm} \chi \right)}{\partial j} + \dfrac{1}{e_1 e_3} \dfrac{\partial \left(A^{lm} \, e_3 \zeta \right)}{\partial i} \end{pmatrix} \end{aligned} \tag{2.43}$$

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

$$\begin{aligned} D_u^{l\mathbf{U}} &= \nabla . \left(\Re \, \nabla u \right) \\ D_v^{l\mathbf{U}} &= \nabla . \left(\Re \, \nabla v \right) \end{aligned} \tag{2.44}$$

where $\Re$ is given by (2.37). It is the same expression as those used for diffusive operator on tracers. It must be emphasised that such a formulation is only exact in a Cartesian coordinate system, $i.e.$ on a $f-$ or $\beta-$plane, not on the sphere. It is also a very good approximation in vicinity of the Equator in a geographical coordinate system [**?**].

lateral fourth order momentum diffusive operator

As for tracers, the fourth order momentum diffusive operator along z or s-surfaces is a re-entering second order operator (2.43) or (2.43) with the eddy viscosity coefficient correctly placed :

geopotential diffusion in z-coordinate :

$$\begin{aligned} \mathbf{D}^{l\mathbf{U}} = & \nabla_h \left\{ \nabla_h . \left[A^{lm} \nabla_h (\chi) \right] \right\} \\ & + \nabla_h \times \left\{ \mathbf{k} \cdot \nabla \times \left[A^{lm} \nabla_h \times (\zeta \, \mathbf{k}) \right] \right\} \end{aligned} \tag{2.45}$$

geopotential diffusion in s-coordinate :

$$\begin{cases} D_u^{l\mathbf{U}} = \Delta \left(A^{lm} \, \Delta u \right) \\ D_v^{l\mathbf{U}} = \Delta \left(A^{lm} \, \Delta v \right) \end{cases} \quad \text{where} \quad \Delta (\bullet) = \nabla \cdot (\Re \nabla (\bullet)) \tag{2.46}$$

3 Time Domain (STP)

Contents

Having defined the continuous equations in Chap. 2, we need now to choose a time discretization. In the present chapter, we provide a general description of the *NEMO* time stepping strategy and the consequences for the order in which the equations are solved.

3.1 Time stepping environment

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

$$x^{t+\Delta t} = x^{t-\Delta t} + 2\,\Delta t\ \mathrm{RHS}_x^{t-\Delta t,\,t,\,t+\Delta t} \tag{3.1}$$

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

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

The three level scheme requires three arrays for each prognostic variable. For each variable x there is x_b (before), x_n (now) and x_a. The third array, although referred to as x_a (after) in the code, is usually not the variable at the after time step ; but rather it is used to store the time derivative (RHS in (3.1)) prior to time-stepping the equation. Generally, the time stepping is performed once at each time step in the *tranxt.F90* and *dynnxt.F90* modules, except when using implicit vertical diffusion or calculating sea surface height in which case time-splitting options are used.

3.2 Non-Diffusive Part — Leapfrog Scheme

The time stepping used for processes other than diffusion is the well-known leapfrog scheme [**?**]. This scheme is widely used for advection processes in low-viscosity fluids. It is a time centred scheme, $i.e.$ the RHS in (3.1) is evaluated at time step t, the now time step. It may be used for momentum and tracer advection, pressure gradient, and Coriolis terms, but not for diffusion terms. 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 diffusion and Rayleigh damping processes. However, the scheme allows the coexistence of a numerical and a physical mode due to its leading third order dispersive error. In other words a divergence of odd and even time steps may occur. To prevent it, the leapfrog scheme is often used in association with a Robert-Asselin time filter (hereafter the LF-RA scheme). This filter, first designed by **?** and more comprehensively studied by **?**, is a kind of laplacian diffusion in time that mixes odd and even time steps :

$$x_F^t = x^t + \gamma \left[x_F^{t-\Delta t} - 2x^t + x^{t+\Delta t} \right] \tag{3.2}$$

where the subscript F denotes filtered values and γ is the Asselin coefficient. γ is initialized as *rn_atfp* (namelist parameter). Its default value is *rn_atfp*=10^{-3} (see § 3.5), causing only a weak dissipation of high frequency motions ([**?**]). The addition of a time filter degrades the accuracy of the calculation from second to first order. However, the second order truncation error is proportional to γ, which is small compared to 1. Therefore, the LF-RA is a quasi second order accurate scheme. The LF-RA scheme is preferred to other time differencing schemes such as predictor corrector or trapezoidal schemes, because the user has an explicit and simple control of the magnitude of the time diffusion of the scheme. When used with the 2nd order space centred discretisation of the advection terms in the momentum and tracer equations, LF-RA avoids implicit numerical diffusion : diffusion is set explicitly by the user through the Robert-Asselin filter parameter and the viscosity and diffusion coefficients.

3.3 Diffusive Part — Forward or Backward Scheme

The leapfrog differencing scheme is unsuitable for the representation of diffusion and damping processes. For a tendancy D_x, representing a diffusion term or a restoring term to a tracer climatology (when present, see § 5.6), a forward time differencing scheme is used :

$$x^{t+\Delta t} = x^{t-\Delta t} + 2\,\Delta t\; {D_x}^{t-\Delta t} \tag{3.3}$$

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

$$A^h < \begin{cases} \dfrac{e^2}{8\,\Delta t} & \text{laplacian diffusion} \\ \dfrac{e^4}{64\,\Delta t} & \text{bilaplacian diffusion} \end{cases} \tag{3.4}$$

where e is the smallest grid size in the two horizontal directions and A^h is the mixing coefficient. The linear constraint (3.4) is a necessary condition, but not sufficient. If it is not satisfied, even mildly, then the model soon becomes wildly unstable. The instability can be removed by either reducing the length of the time steps or reducing the mixing coefficient.

For the vertical diffusion terms, a forward time differencing scheme can be used, but usually the numerical stability condition imposes a strong constraint on the time step. Two

solutions are available in *NEMO* to overcome the stability constraint : (a) a forward time differencing scheme using a time splitting technique (*ln_zdfexp* = true) or (b) a backward (or implicit) time differencing scheme (*ln_zdfexp* = false). In (a), the master time step Δt is cut into N fractional time steps so that the stability criterion is reduced by a factor of N. The computation is performed as follows :

$$\begin{aligned} x_*^{t-\Delta t} &= x^{t-\Delta t} \\ x_*^{t-\Delta t+L\frac{2\Delta t}{N}} &= x_*^{t-\Delta t+(L-1)\frac{2\Delta t}{N}} + \frac{2\Delta t}{N}\,\mathrm{DF}^{t-\Delta t+(L-1)\frac{2\Delta t}{N}} \quad \text{for } L=1 \text{ to } N \\ x^{t+\Delta t} &= x_*^{t+\Delta t} \end{aligned} \tag{3.5}$$

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

$$x^{t+\Delta t} = x^{t-\Delta t} + 2\,\Delta t\,\mathrm{RHS}_x^{t+\Delta t} \tag{3.6}$$

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

$$\frac{T(k)^{t+1}-T(k)^{t-1}}{2\,\Delta t} \equiv \mathrm{RHS} + \frac{1}{e_{3t}}\delta_k\left[\frac{A_w^{vT}}{e_{3w}}\delta_{k+1/2}\left[T^{t+1}\right]\right] \tag{3.7}$$

where RHS is the right hand side of the equation except for the vertical diffusion term. We rewrite (3.6) as :

$$-c(k+1)\,T^{t+1}(k+1) + d(k)\,T^{t+1}(k) - \,c(k)\,T^{t+1}(k-1) \equiv b(k) \tag{3.8}$$

where

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

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

3.4 Hydrostatic Pressure Gradient — semi-implicit scheme

The range of stability of the Leap-Frog scheme can be extended by a factor of two by introducing a semi-implicit computation of the hydrostatic pressure gradient term [**?**]. Instead of evaluating the pressure at t, a linear combination of values at $t-\Delta t$, t and

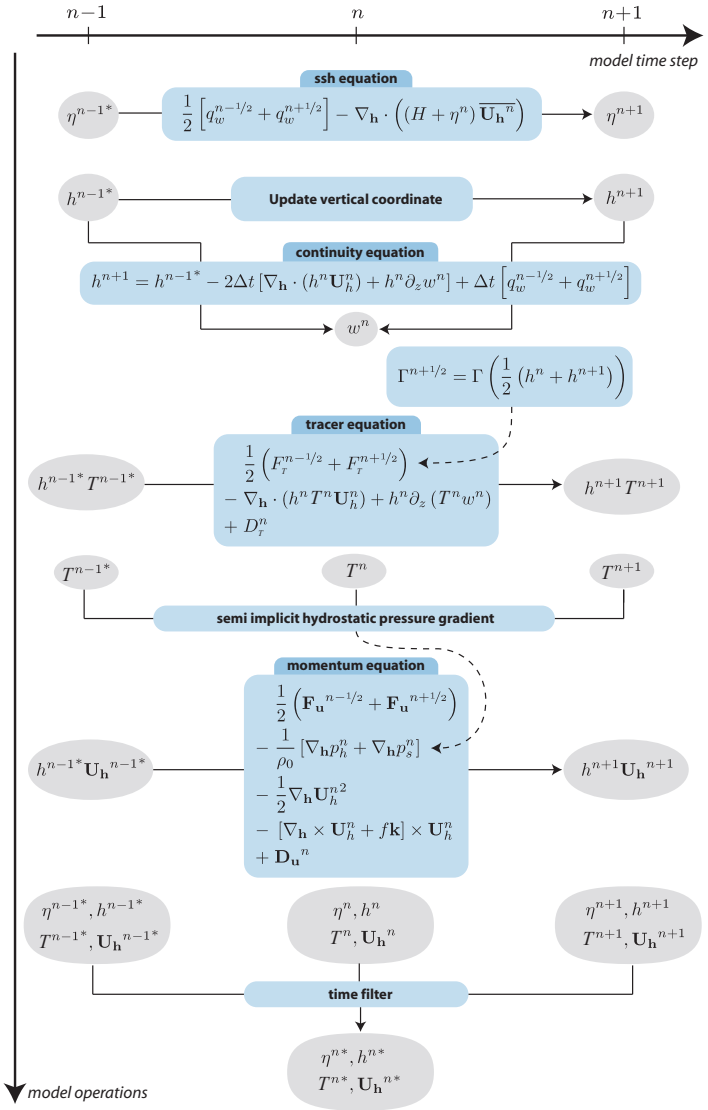

FIG. 3.1 – Sketch of the leapfrog time stepping sequence in *NEMO* from **?**. The use of a semi-implicit computation of the hydrostatic pressure gradient requires the tracer equation to be stepped forward prior to the momentum equation. The need for knowledge of the vertical scale factor (here denoted as h) requires the sea surface height and the continuity equation to be stepped forward prior to the computation of the tracer equation. Note that the method for the evaluation of the surface pressure gradient ∇p_s is not presented here (see § 6.5).

$t + \Delta t$ is used (see § 6.4.4). This technique, controlled by the *nn_dynhpg_rst* namelist

parameter, does not introduce a significant additional computational cost when tracers and thus density is time stepped before the dynamics. This time step ordering is used in *NEMO* (Fig.3.1).

This technique, used in several GCMs (*NEMO* , POP or MOM for instance), makes the Leap-Frog scheme as efficient [1] as the Forward-Backward scheme used in MOM [**?**] and more efficient than the LF-AM3 scheme (leapfrog time stepping combined with a third order Adams-Moulthon interpolation for the predictor phase) used in ROMS [**?**].

In fact, this technique is efficient when the physical phenomenon that limits the time-step is internal gravity waves (IGWs). Indeed, it is equivalent to applying a time filter to the pressure gradient to eliminate high frequency IGWs. Obviously, the doubling of the time-step is achievable only if no other factors control the time-step, such as the stability limits associated with advection, diffusion or Coriolis terms. For example, it is inefficient in low resolution global ocean configurations, since inertial oscillations in the vicinity of the North Pole are the limiting factor for the time step. It is also often inefficient in very high resolution configurations where strong currents and small grid cells exert the strongest constraint on the time step.

3.5 The Modified Leapfrog – Asselin Filter scheme

Significant changes have been introduced by **?** in the LF-RA scheme in order to ensure tracer conservation and to allow the use of a much smaller value of the Asselin filter parameter. The modifications affect both the forcing and filtering treatments in the LF-RA scheme.

In a classical LF-RA environment, the forcing term is centred in time, $i.e.$ it is time-stepped over a $2\Delta t$ period : $x^t = x^t + 2\Delta t Q^t$ where Q is the forcing applied to x, and the time filter is given by (3.2) so that Q is redistributed over several time step. In the modified LF-RA environment, these two formulations have been replaced by :

$$x^{t+\Delta t} = x^{t-\Delta t} + \Delta t \left(Q^{t-\Delta t/2} + Q^{t+\Delta t/2} \right) \tag{3.9}$$

$$x_F^t = x^t + \gamma \left[x_F^{t-\Delta t} - 2x^t + x^{t+\Delta t} \right] - \gamma \, \Delta t \left[Q^{t+\Delta t/2} - Q^{t-\Delta t/2} \right] \tag{3.10}$$

The change in the forcing formulation given by (3.9) (see Fig.3.2) has a significant effect : the forcing term no longer excites the divergence of odd and even time steps [**?**]. This property improves the LF-RA scheme in two respects. First, the LF-RA can now ensure the local and global conservation of tracers. Indeed, time filtering is no longer required on the forcing part. The influence of the Asselin filter on the forcing is be removed by adding a new term in the filter (last term in (3.10) compared to (3.2)). Since the filtering of the forcing was the source of non-conservation in the classical LF-RA scheme, the modified formulation becomes conservative [**?**]. Second, the LF-RA becomes a truly quasi-second

[1] The efficiency is defined as the maximum allowed Courant number of the time stepping scheme divided by the number of computations of the right-hand side per time step.

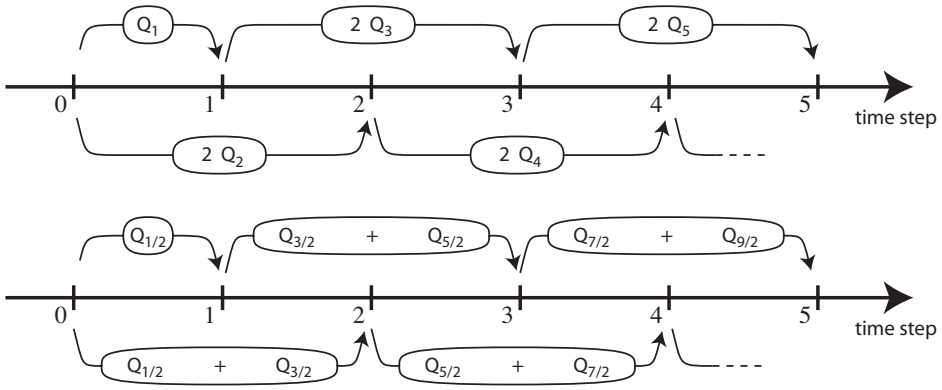

FIG. 3.2 – Illustration of forcing integration methods. (top) "Traditional" formulation : the forcing is defined at the same time as the variable to which it is applied (integer value of the time step index) and it is applied over a $2\Delta t$ period. (bottom) modified formulation : the forcing is defined in the middle of the time (integer and a half value of the time step index) and the mean of two successive forcing values $(n - 1/2, n + 1/2)$. is applied over a $2\Delta t$ period.

order scheme. Indeed, (3.9) used in combination with a careful treatment of static instability (§10.2.2) and of the TKE physics (§10.1.4), the two other main sources of time step divergence, allows a reduction by two orders of magnitude of the Asselin filter parameter.

Note that the forcing is now provided at the middle of a time step : $Q^{t+\Delta t/2}$ is the forcing applied over the $[t, t + \Delta t]$ time interval. This and the change in the time filter, (3.10), allows an exact evaluation of the contribution due to the forcing term between any two time steps, even if separated by only Δt since the time filter is no longer applied to the forcing term.

3.6 Start/Restart strategy

```
!-----------------------------------------------------------------------
&namrun        !   parameters of the run
!-----------------------------------------------------------------------
   nn_no       =       0   !  job number
   cn_exp      =  "ORCA2"  !  experience name
   nn_it000    =       1   !  first time step
   nn_itend    =    5475   !  last  time step (std 5475)
   nn_date0    =  010101   !  initial calendar date yymmdd (used if nn_rstctl=1)
   nn_leapy    =       0   !  Leap year calendar (1) or not (0)
   ln_rstart   = .false.   !  start from rest (F) or from a restart file (T)
   nn_rstctl   =       0       !  restart control = 0 nn_it000 is not compared to the restart file value
                               !                  = 1 use nn_date0 in namelist (not the value in the restart file)
                               !                  = 2 calendar parameters read in the restart file
   cn_ocerst_in  = "restart"   !  suffix of ocean restart name (input)
   cn_ocerst_out = "restart"   !  suffix of ocean restart name (output)
   nn_istate   =       0   !  output the initial state (1) or not (0)
   nn_stock    =    5475   !  frequency of creation of a restart file (modulo referenced to 1)
   nn_write    =    5475   !  frequency of write in the output file   (modulo referenced to nn_it000)
   ln_dimgnnn  = .false.   !  DIMG file format: 1 file for all processors (F) or by processor (T)
   ln_mskland  = .false.   !  mask land points in NetCDF outputs (costly: + ~15%)
   ln_clobber  = .false.   !  clobber (overwrite) an existing file
   nn_chunksz  =       0   !  chunksize (bytes) for NetCDF file (works only with iom_nf90 routines)
```

/

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

$$x^1 = x^0 + \Delta t \, \text{RHS}^0 \tag{3.11}$$

This is done simply by keeping the leapfrog environment ($i.e.$ the (3.1) three level time stepping) but setting all x^0 (*before*) and x^1 (*now*) fields equal at the first time step and using half the value of Δt.

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

Note that when a semi-implicit scheme is used to evaluate the hydrostatic pressure gradient (see §6.4.4), an extra three-dimensional field has to be added to the restart file to ensure an exact restartability. This is done optionally via the *nn_dynhpg_rst* namelist parameter, so that the size of the restart file can be reduced when restartability is not a key issue (operational oceanography or in ensemble simulations for seasonal forecasting).

Note the size of the time step used, Δt, is also saved in the restart file. When restarting, if the the time step has been changed, a restart using an Euler time stepping scheme is imposed.

4 Space Domain (DOM)

Contents

Having defined the continuous equations in Chap. 2 and chosen a time discretization Chap. 3, we need to choose a discretization on a grid, and numerical algorithms. In the present chapter, we provide a general description of the staggered grid used in *NEMO* , and other information relevant to the main directory routines as well as the DOM (DOMain) directory.

4.1 Fundamentals of the Discretisation

4.1.1 Arrangement of Variables

The numerical techniques used to solve the Primitive Equations in this model are based on the traditional, centred second-order finite difference approximation. Special attention has been given to the homogeneity of the solution in the three space directions. The arrangement of variables is the same in all directions. It consists of cells centred on scalar points (t, S, p, ρ) with vector points (u, v, w) defined in the centre of each face of the cells (Fig. 4.1). 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 4.1. 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 (2.10). As a result, the mesh on which partial derivatives $\frac{\partial}{\partial\lambda}$, $\frac{\partial}{\partial\varphi}$, and $\frac{\partial}{\partial z}$ are evaluated is a uniform mesh with a grid size of unity. Discrete partial derivatives are formulated by the traditional, centred second order finite difference approximation while the scale factors are chosen equal to their local analytical value. An important point here is that the partial derivative of the scale factors must be evaluated by centred finite difference approximation, not from their analytical expression. This preserves the symmetry of the discrete set of equations and therefore satisfies many of the continuous properties (see Appendix E). 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 (4.8)) in the next section).

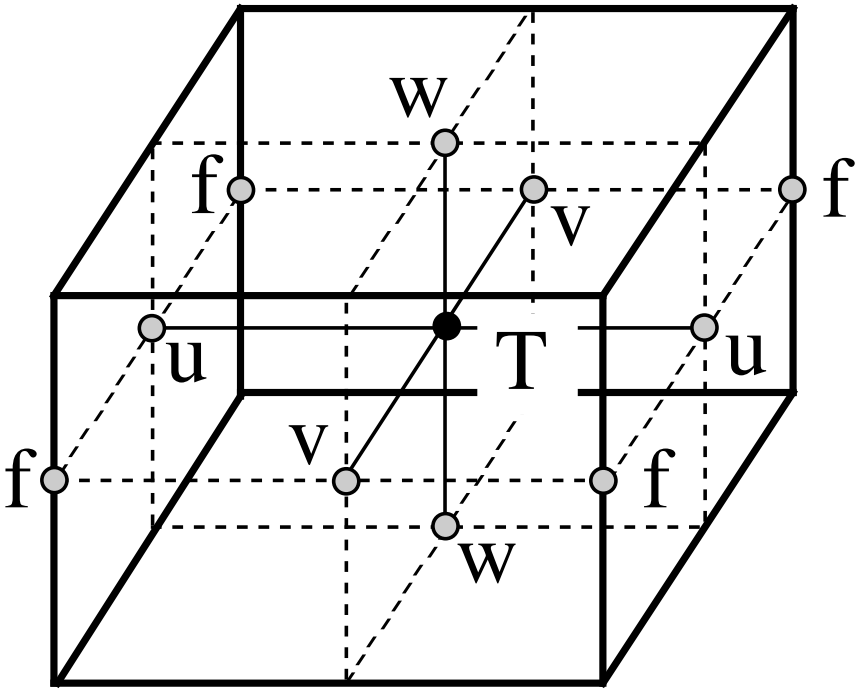

FIG. 4.1 – Arrangement of variables. t indicates scalar points where temperature, salinity, density, pressure and horizontal divergence are defined. (u,v,w) indicates vector points, and f indicates vorticity points where both relative and planetary vorticities are defined

4.1.2 Discrete Operators

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

$$\delta_i[q] = \quad q(i+1/2) - q(i-1/2) \tag{4.1a}$$

$$\overline{q}^{\,i} = \{q(i+1/2) + q(i-1/2)\} \;/\; 2 \tag{4.1b}$$

Similar operators are defined with respect to $i+1/2$, j, $j+1/2$, k, and $k+1/2$. Following (2.11a) and (2.11d), the gradient of a variable q defined at a t-point has its three components defined at u-, v- and w-points while its Laplacien is defined at t-point.

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

TAB. 4.1 – Location of grid-points as a function of integer or integer and a half value of the column, line or level. This indexing is only used for the writing of the semi-discrete equation. In the code, the indexing uses integer values only and has a reverse direction in the vertical (see §4.1.3)

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

$$\nabla q \equiv \frac{1}{e_{1u}} \delta_{i+1/2}[q] \;\mathbf{i} + \frac{1}{e_{2v}} \delta_{j+1/2}[q] \;\mathbf{j} + \frac{1}{e_{3w}} \delta_{k+1/2}[q] \;\mathbf{k} \tag{4.2}$$

$$\Delta q \equiv \frac{1}{e_{1t}\, e_{2t}\, e_{3t}} \left(\delta_i \left[\frac{e_{2u}\, e_{3u}}{e_{1u}} \, \delta_{i+1/2}[q] \right] + \delta_j \left[\frac{e_{1v}\, e_{3v}}{e_{2v}} \, \delta_{j+1/2}[q] \right] \right) + \frac{1}{e_{3t}} \delta_k \left[\frac{1}{e_{3w}} \, \delta_{k+1/2}[q] \right] \tag{4.3}$$

Following (2.11c) and (2.11b), a vector $\mathbf{A} = (a_1, a_2, a_3)$ defined at vector points (u, v, w) has its three curl components defined at vw-, uw, and f-points, and its divergence defined at t-points :

$$\nabla \times \mathbf{A} \equiv \frac{1}{e_{2v}\, e_{3vw}} \left(\delta_{j+1/2} \left[e_{3w}\, a_3 \right] - \delta_{k+1/2} \left[e_{2v}\, a_2 \right] \right) \mathbf{i} \tag{4.4}$$

$$+ \frac{1}{e_{2u}\, e_{3uw}} \left(\delta_{k+1/2} \left[e_{1u}\, a_1 \right] - \delta_{i+1/2} \left[e_{3w}\, a_3 \right] \right) \mathbf{j} \tag{4.5}$$

$$+ \frac{1}{e_{1f}\, e_{2f}} \left(\delta_{i+1/2} \left[e_{2v}\, a_2 \right] - \delta_{j+1/2} \left[e_{1u}\, a_1 \right] \right) \mathbf{k} \tag{4.6}$$

$$\nabla \cdot \mathbf{A} = \frac{1}{\mathrm{e_{1t}\, e_{2t}\, e_{3t}}} \left(\delta_{\mathrm{i}} \left[\mathrm{e_{2u}\, e_{3u}\, a_1} \right] + \delta_{\mathrm{j}} \left[\mathrm{e_{1v}\, e_{3v}\, a_2} \right] \right) + \frac{1}{\mathrm{e_{3t}}} \delta_{\mathrm{k}} \left[\mathrm{a_3} \right] \tag{4.7}$$

In the special case of a pure z-coordinate system, (4.3) and (4.7) can be simplified. In this case, the vertical scale factor becomes a function of the single variable k and thus does not depend on the horizontal location of a grid point. For example (4.7) reduces to :

$$\nabla \cdot \mathbf{A} = \frac{1}{\mathrm{e_{1t}\, e_{2t}}} \left(\delta_{\mathrm{i}} \left[\mathrm{e_{2u}\, a_1} \right] + \delta_{\mathrm{j}} \left[\mathrm{e_{1v}\, a_2} \right] \right) + \frac{1}{\mathrm{e_{3t}}} \delta_{\mathrm{k}} \left[\mathrm{a_3} \right]$$

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

$$\bar{q} = \frac{1}{H} \int_{k^b}^{k^o} q \; e_{3q} \, dk \equiv \frac{1}{H_q} \sum_k q \; e_{3q} \tag{4.8}$$

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

In continuous form, the following properties are satisfied :

$$\nabla \times \nabla q = \mathbf{0} \tag{4.9}$$

$$\nabla \cdot (\nabla \times \mathbf{A}) = 0 \tag{4.10}$$

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

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

$$\sum_i a_i \; \delta_i \left[b \right] \equiv - \sum_i \delta_{i+1/2} \left[a \right] \; b_{i+1/2} \tag{4.11}$$

$$\sum_i a_i \; \overline{b}^{\,i} \equiv \sum_i \overline{a}^{\,i+1/2} \; b_{i+1/2} \tag{4.12}$$

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

4.1.3 Numerical Indexing

The array representation used in the FORTRAN code requires an integer indexing while the analytical definition of the mesh (see §4.1.1) is associated with the use of integer values for t-points and both integer and integer and a half values for all the other points. Therefore a specific integer indexing must be defined for points other than t-points ($i.e.$ velocity and vorticity grid-points). Furthermore, the direction of the vertical indexing has been changed so that the surface level is at $k = 1$.

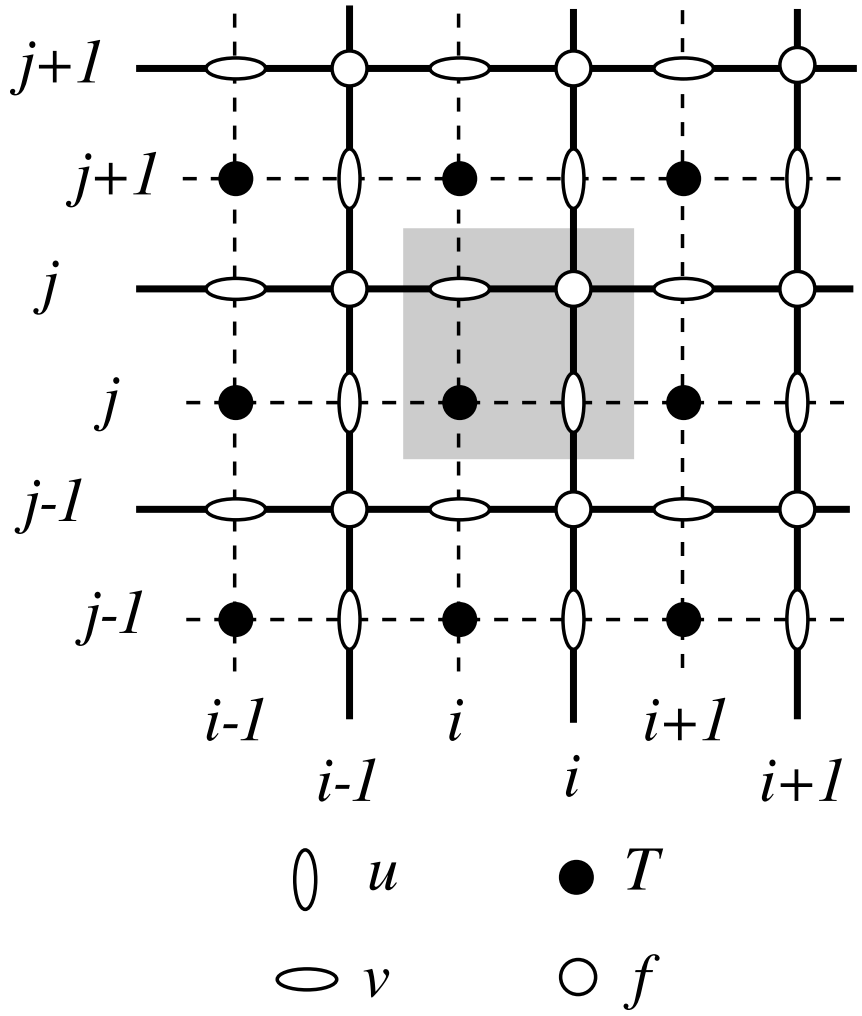

FIG. 4.2 – Horizontal integer indexing used in the FORTRAN code. The dashed area indicates the cell in which variables contained in arrays have the same i- and j-indices

Horizontal Indexing

The indexing in the horizontal plane has been chosen as shown in Fig.4.2. 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.4.2). A t-point and its nearest northeast f-point have the same i-and j-indices.

Vertical Indexing

In the vertical, the chosen indexing requires special attention since the k-axis is re-orientated downward in the FORTRAN code compared to the indexing used in the semi-discrete equations and given in §4.1.1. The sea surface corresponds to the w-level $k = 1$ which is the same index as t-level just below (Fig.4.3). The last w-level ($k = jpk$) either corresponds to the ocean floor or is inside the bathymetry while the last t-level is always inside the bathymetry (Fig.4.3). Note that for an increasing k index, a w-point and the t-point just below have the same k index, in opposition to what is done in the horizontal plane where it is the t-point and the nearest velocity points in the direction of the horizontal axis that have the same i or j index (compare the dashed area in Fig.4.2 and 4.3). Since the scale factors are chosen to be strictly positive, a *minus sign* appears in the FORTRAN code *before all the vertical derivatives* of the discrete equations given in this documentation.

Domain Size

The total size of the computational domain is set by the parameters *jpiglo*, *jpjglo* and *jpk* in the i, j and k directions respectively. They are given as parameters in the *par_oce.F90* module[1]. The use of parameters rather than variables (together with dynamic allocation of arrays) was chosen because it ensured that the compiler would optimize the executable code efficiently, especially on vector machines (optimization may be less efficient when the problem size is unknown at the time of compilation). Nevertheless, it is possible to set up the code with full dynamical allocation by using the AGRIF packaged [**?**]. Note that are other parameters in *par_oce.F90* that refer to the domain size. The two parameters $jpidta$ and $jpjdta$ may be larger than $jpiglo$, $jpjglo$ when the user wants to use only a sub-region of a given configuration. This is the "zoom" capability described in §13.3. 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 §8.3).

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

4.2.1 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 4.1. The associated scale

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

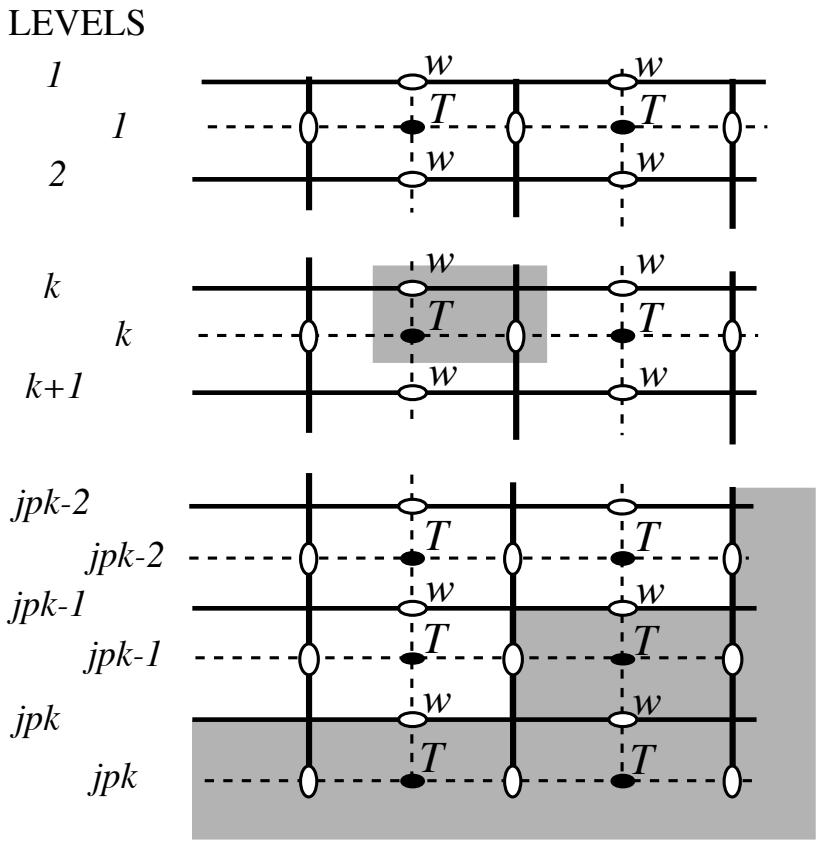

FIG. 4.3 – Vertical integer indexing used in the FORTRAN code. Note that the k-axis is orientated downward. The dashed area indicates the cell in which variables contained in arrays have the same k-index.

factors are defined using the analytical first derivative of the transformation (2.10). 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 longitude λ and latitude φ as a function of (i, j). The horizontal scale factors are calculated using (2.10). For example, when the longitude and latitude are function of a single value (i and j, respectively) (geographical configuration of the mesh), the horizontal mesh definition reduces to define the wanted $\lambda(i)$, $\varphi(j)$, and their derivatives $\lambda'(i)$ $\varphi'(j)$ in the *domhgr.F90* module. The model computes the grid-point positions and scale factors in the horizontal plane as follows :

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

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

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

Note that the definition of the scale factors ($i.e.$ as the analytical first derivative of the transformation that gives (λ, φ, z) as a function of (i, j, k)) is specific to the *NEMO* model [**?**]. As an example, e_{1t} is defined locally at a t-point, whereas many other models on a C grid choose to define such a scale factor as the distance between the U-points on each side of the t-point. Relying on an analytical transformation has two advantages : firstly, there is no ambiguity in the scale factors appearing in the discrete equations, since they are first introduced in the continuous equations ; secondly, analytical transformations encourage good practice by the definition of smoothly varying grids (rather than allowing the user to set arbitrary jumps in thickness between adjacent layers) [**?**]. An example of the effect of such a choice is shown in Fig. 4.4.

4.2.2 Choice of horizontal grid

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

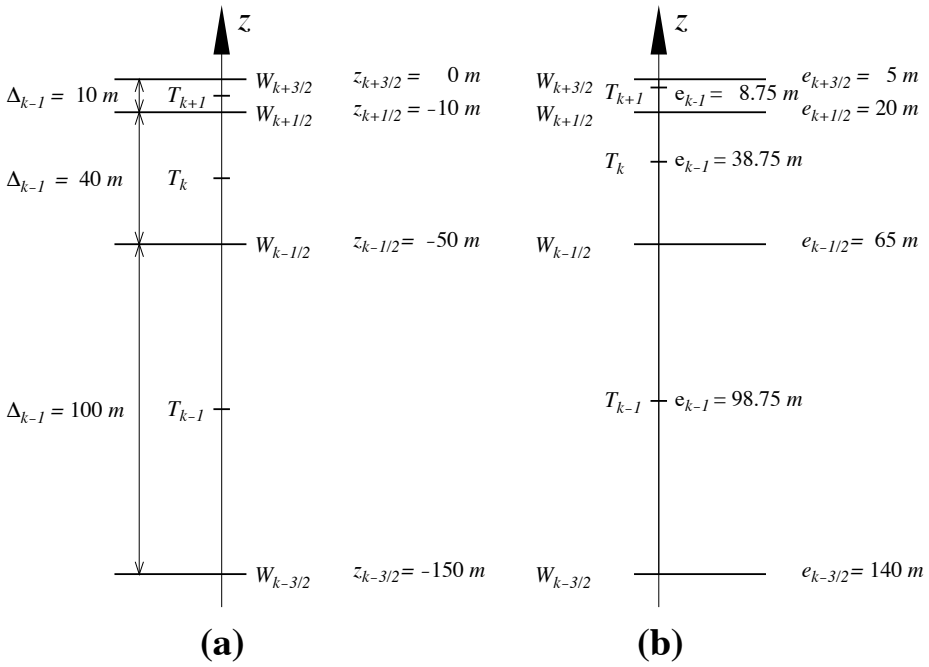

FIG. 4.4 – Comparison of (a) traditional definitions of grid-point position and grid-size in the vertical, and (b) analytically derived grid-point position and scale factors. For both grids here, the same w-point depth has been chosen but in (a) the t-points are set half way between w-points while in (b) they are defined from an analytical function : $z(k) = 5\,(i-1/2)^3 - 45\,(i-1/2)^2 + 140\,(i-1/2) - 150$. Note the resulting difference between the value of the grid-size Δ_k and those of the scale factor e_k.

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

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

There are two simple cases of geographical grids on the sphere. With *jphgr_mesh*=1, the grid (expressed in degrees) is regular in space, with grid sizes specified by parameters *ppe1_deg* and *ppe2_deg*, respectively. Such a geographical grid can be very anisotropic at high latitudes because of the convergence of meridians (the zonal scale factors e_1 become much smaller than the meridional scale factors e_2). The Mercator grid (*jphgr_mesh*=4) avoids this anisotropy by refining the meridional scale factors in the same way as the zonal ones. In this case, meridional scale factors and latitudes are calculated analytically using the formulae appropriate for a Mercator projection, based on *ppe1_deg* which is a

reference grid spacing at the equator (this applies even when the geographical equator is situated outside the model domain). In these two cases (*jphgr_mesh*=1 or 4), the grid position is defined by the longitude and latitude of the south-westernmost point (*ppglamt0* and *ppgphi0*). Note that for the Mercator grid the user need only provide an approximate starting latitude : the real latitude will be recalculated analytically, in order to ensure that the equator corresponds to line passing through t- and u-points.

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

The choice of the grid must be consistent with the boundary conditions specified by the parameter *jperio* (see §8).

4.2.3 Output Grid files

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

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

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


!-----------------------------------------------------------------------
&namdom        !   space and time domain (bathymetry, mesh, timestep)
!-----------------------------------------------------------------------
```

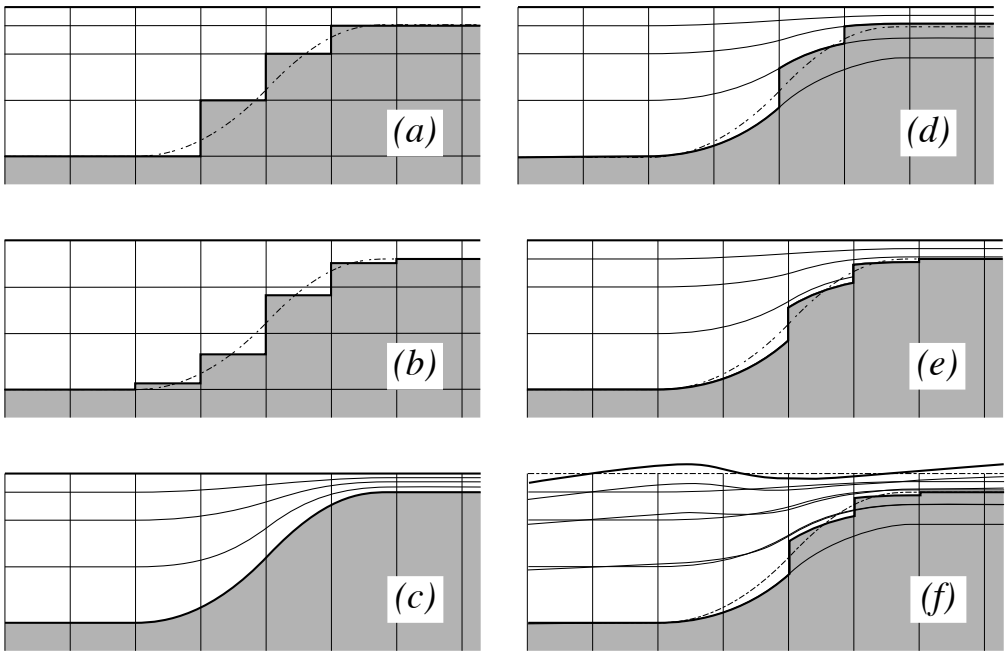

FIG. 4.5 – The ocean bottom as seen by the model : (a) z-coordinate with full step, (b) z-coordinate with partial step, (c) s-coordinate : terrain following representation, (d) hybrid $s-z$ coordinate, (e) hybrid $s-z$ coordinate with partial step, and (f) same as (e) but with variable volume associated with the non-linear free surface. Note that the variable volume option (**key_vvl**) can be used with any of the 5 coordinates (a) to (e).

```
   nn_bathy    =    1      !  compute (=0) or read (=1) the bathymetry file
   nn_closea    =   0      !  remove (=0) or keep (=1) closed seas and lakes (ORCA)
   nn_msh      =    0      !  create (=1) a mesh file or not (=0)
   rn_hmin     =   -3.     !  min depth of the ocean (>0) or min number of ocean level (<0)
   rn_e3zps_min=   20.     !  partial step thickness is set larger than the minimum of
   rn_e3zps_rat=    0.1    !  rn_e3zps_min and rn_e3zps_rat*e3t, with 0<rn_e3zps_rat<1
                           !
   rn_rdt      = 5760.     !  time step for the dynamics (and tracer if nn_acc=0)
   nn_baro     =   64      !  number of barotropic time step            ("key_dynspg_ts")
   rn_atfp     =    0.1    !  asselin time filter parameter
   nn_acc      =    0      !  acceleration of convergence : =1      used, rdt < rdttra(k)
                           !                                =0, not used, rdt = rdttra
   rn_rdtmin   = 28800.          !  minimum time step on tracers (used if nn_acc=1)
   rn_rdtmax   = 28800.          !  maximum time step on tracers (used if nn_acc=1)
   rn_rdth     =  800.           !  depth variation of tracer time step  (used if nn_acc=1)
/
```

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

The choice of a vertical coordinate, even if it is made through a namelist parameter, must be done once of all at the beginning of an experiment. It is not intended as

an option which can be enabled or disabled in the middle of an experiment. Three main choices are offered (Fig. 4.5a to c) : z-coordinate with full step bathymetry (*ln_zco* = true), z-coordinate with partial step bathymetry (*ln_zps* = true), or generalized, s-coordinate (*ln_sco* = true). Hybridation of the three main coordinates are available : $s-z$ or $s-zps$ coordinate (Fig. 4.5d and 4.5e). When using the variable volume option **key_vvl**) ($i.e.$ non-linear free surface), the coordinate follow the time-variation of the free surface so that the transformation is time dependent : $z(i,j,k,t)$ (Fig. 4.5f). This option can be used with full step bathymetry or s-coordinate (hybride and partial step coordinates have not yet been tested in NEMO v2.3).

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

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

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

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

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

The arrays describing the grid point depths and vertical scale factors are three dimensional arrays (i,j,k) even in the case of z-coordinate with full step bottom topography. In non-linear free surface (**key_vvl**), their knowledge is required at *before*, *now* and *after* time step, while they do not vary in time in linear free surface case. To improve the code readability while providing this flexibility, the vertical coordinate and scale factors are defined as functions of (i,j,k) with "fs" as prefix (examples : *fse3t_b, fse3t_n, fse3t_a,* for the *before*, *now* and *after* scale factors at t-point) that can be either three different arrays when **key_vvl** is defined, or a single fixed arrays. These functions are defined in the file *domzgr_substitute.h90* of the DOM directory. They are used throughout the code, and replaced by the corresponding arrays at the time of pre-processing (CPP capability).

4.3.1 Meter Bathymetry

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

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

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

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

***nn_bathy* = 1** read a bathymetry. The *bathy_meter.nc* 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. Defining the bathymetry also defines the coastline : where the bathymetry is zero, no model levels are defined (all levels are masked).

When a global ocean is coupled to an atmospheric model it is better to represent all large water bodies (e.g, great lakes, Caspian sea...) even if the model resolution does not allow their communication with the rest of the ocean. This is unnecessary when the ocean is forced by fixed atmospheric conditions, so these seas can be removed from the ocean domain. The user has the option to set the bathymetry in closed seas to zero (see §13.2), but the code has to be adapted to the user's configuration.

4.3.2 z-coordinate (*ln_zco*=true) and reference coordinate

The reference coordinate transformation $z_0(k)$ defines the arrays $gdept_0$ and $gdepw_0$ for t- and w-points, respectively. As indicated on Fig.4.3 *jpk* is the number of w-levels. $gdepw_0(1)$ is the ocean surface. There are at most *jpk*-1 t-points inside the ocean, the additional t-point at $jk = jpk$ is below the sea floor and is not used. The vertical location of w- and t-levels is defined from the analytic expression of the depth $z_0(k)$ whose analytical derivative with respect to k provides the vertical scale factors. The user must provide the analytical expression of both z_0 and its first derivative with respect to k. This is done in routine *domzgr.F90* through statement functions, using parameters provided in the *par_oce.h90* file.

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

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

$$\begin{aligned} z_0(k) &= h_{sur} - h_0\, k - h_1\, \log\left[\cosh\left((k-h_{th})/h_{cr}\right)\right] \\ e_3^0(k) &= \left|-h_0 - h_1\, \tanh\left((k-h_{th})/h_{cr}\right)\right| \end{aligned} \tag{4.13}$$

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. 4.6).

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

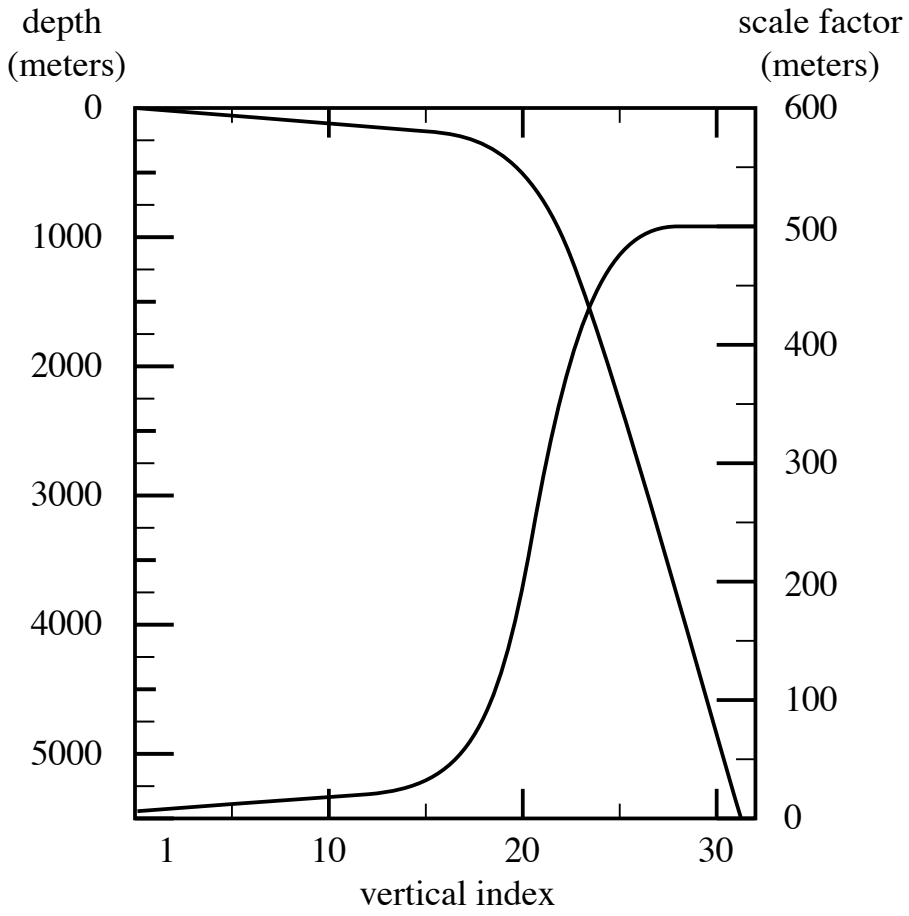

FIG. 4.6 – Default vertical mesh for ORCA2 : 30 ocean levels (L30). Vertical level functions for (a) T-point depth and (b) the associated scale factor as computed from (4.13) using (4.14) in z-coordinate.

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

$$\begin{aligned} e_3(1+1/2) &= 10. \\ e_3(jpk-1/2) &= 500. \\ z(1) &= 0. \\ z(jpk) &= -5000. \end{aligned} \tag{4.14}$$

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 (4.13) have been determined such that (4.14) is satisfied, through an optimisation procedure using a bisection method. For the first standard

ORCA2 vertical grid this led to the following values : $h_{sur} = 4762.96$, $h_0 = 255.58$, $h_1 = 245.5813$, and $h_{th} = 21.43336$. The resulting depths and scale factors as a function of the model levels are shown in Fig. 4.6 and given in Table 4.2. Those values correspond to the parameters *ppsur*, *ppa0*, *ppa1*, *ppkth* in the parameter file *par_oce.F90*.

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

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

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

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

```
!-----------------------------------------------------------------------
&namdom        !   space and time domain (bathymetry, mesh, timestep)
!-----------------------------------------------------------------------
   nn_bathy    =    1      !  compute (=0) or read (=1) the bathymetry file
   nn_closea    =   0      !  remove (=0) or keep (=1) closed seas and lakes (ORCA)
   nn_msh      =    0      !  create (=1) a mesh file or not (=0)
   rn_hmin     =   -3.     !  min depth of the ocean (>0) or min number of ocean level (<0)
   rn_e3zps_min=   20.     !  partial step thickness is set larger than the minimum of
   rn_e3zps_rat=    0.1    !  rn_e3zps_min and rn_e3zps_rat*e3t, with 0<rn_e3zps_rat<1
                           !
   rn_rdt      = 5760.     !  time step for the dynamics (and tracer if nn_acc=0)
   nn_baro     =   64      !  number of barotropic time step            ("key_dynspg_ts")
   rn_atfp     =    0.1    !  asselin time filter parameter
   nn_acc      =    0      !  acceleration of convergence : =1      used, rdt < rdttra(k)
                                  !                          =0, not used, rdt = rdttra
   rn_rdtmin   = 28800.          !  minimum time step on tracers (used if nn_acc=1)
   rn_rdtmax   = 28800.          !  maximum time step on tracers (used if nn_acc=1)
   rn_rdth     =  800.           !  depth variation of tracer time step  (used if nn_acc=1)
/
```

In z-coordinate partial step, the depths of the model levels are defined by the reference analytical function $z_0(k)$ as described in the previous section, *except* in the bottom layer. The thickness of the bottom layer is allowed to vary as a function of geographical location (λ, φ) to allow a better representation of the bathymetry, especially in the case of small slopes (where the bathymetry varies by less than one level thickness from one grid point to the next). The reference layer thicknesses e^0_{3t} have been defined in the absence of bathymetry. With partial steps, layers from 1 to *jpk*-2 can have a thickness smaller than $e_{3t}(jk)$. The model deepest layer (*jpk*-1) is allowed to have either a smaller or larger thickness than $e_{3t}(jpk)$: the maximum thickness allowed is $2 * e_{3t}(jpk-1)$. This has to be kept in mind when specifying the maximum depth *pphmax* in partial steps : for example, with *pphmax*$= 5750\ m$ for the DRAKKAR 45 layer grid, the maximum ocean depth allowed is actually $6000\ m$ (the default thickness $e_{3t}(jpk-1)$ being $250\ m$). Two variables in the namdom namelist are used to define the partial step vertical grid. The mimimum water thickness (in meters) allowed for a cell partially filled with bathymetry at level jk is the minimum of *rn_e3zps_min* (thickness in meters, usually $20\ m$) or $e_{3t}(jk) *$ *rn_e3zps_rat* (a fraction, usually 10%, of the default thickness $e_{3t}(jk)$).

LEVEL	gdept	gdepw	e3t	e3w
1	**5.00**	0.00	**10.00**	10.00
2	**15.00**	10.00	**10.00**	10.00
3	**25.00**	20.00	**10.00**	10.00
4	**35.01**	30.00	**10.01**	10.00
5	**45.01**	40.01	**10.01**	10.01
6	**55.03**	50.02	**10.02**	10.02
7	**65.06**	60.04	**10.04**	10.03
8	**75.13**	70.09	**10.09**	10.06
9	**85.25**	80.18	**10.17**	10.12
10	**95.49**	90.35	**10.33**	10.24
11	**105.97**	100.69	**10.65**	10.47
12	**116.90**	111.36	**11.27**	10.91
13	**128.70**	122.65	**12.47**	11.77
14	**142.20**	135.16	**14.78**	13.43
15	**158.96**	150.03	**19.23**	16.65
16	**181.96**	169.42	**27.66**	22.78
17	**216.65**	197.37	**43.26**	34.30
18	**272.48**	241.13	**70.88**	55.21
19	**364.30**	312.74	**116.11**	90.99
20	**511.53**	429.72	**181.55**	146.43
21	**732.20**	611.89	**261.03**	220.35
22	**1033.22**	872.87	**339.39**	301.42
23	**1405.70**	1211.59	**402.26**	373.31
24	**1830.89**	1612.98	**444.87**	426.00
25	**2289.77**	2057.13	**470.55**	459.47
26	**2768.24**	2527.22	**484.95**	478.83
27	**3257.48**	3011.90	**492.70**	489.44
28	**3752.44**	3504.46	**496.78**	495.07
29	**4250.40**	4001.16	**498.90**	498.02
30	**4749.91**	4500.02	**500.00**	499.54
31	**5250.23**	5000.00	**500.56**	500.33

TAB. 4.2 – Default vertical mesh in z-coordinate for 30 layers ORCA2 configuration as computed from (4.13) using the coefficients given in (4.14)

Add a figure here of pstep especially at last ocean level

4.3.4 s-coordinate (*ln_sco*=true)

```
!-----------------------------------------------------------------------
&namzgr_sco    !   s-coordinate or hybrid z-s-coordinate
!-----------------------------------------------------------------------
   rn_sbot_min =  300.     !  minimum depth of s-bottom surface (>0) (m)
   rn_sbot_max = 5250.     !  maximum depth of s-bottom surface (= ocean depth) (>0) (m)
   rn_theta    =    6.0    !  surface control parameter (0<=rn_theta<=20)
   rn_thetb    =    0.75   !  bottom control parameter  (0<=rn_thetb<= 1)
   rn_rmax     =    0.15   !  maximum cut-off r-value allowed (0<rn_max<1)
   ln_s_sigma  = .false.   !  hybrid s-sigma coordinates
   rn_bb       =    0.8    !  stretching with s-sigma
   rn_hc       =  150.0    !  critical depth with s-sigma
/
```

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

$$\begin{aligned} z(k) &= h(i,j)\ z_0(k) \\ e_3(k) &= h(i,j)\ z'_0(k) \end{aligned} \tag{4.15}$$

where h is the depth of the last w-level ($z_0(k)$) defined at the t-point location in the horizontal and $z_0(k)$ is a function which varies from 0 at the sea surface to 1 at the ocean bottom. The depth field h is not necessary the ocean depth, since a mixed step-like and bottom-following representation of the topography can be used (Fig. 4.5d-e). In the example provided (*zgr_sco* routine, see *domzgr.F90*) h is a smooth envelope bathymetry and steps are used to represent sharp bathymetric gradients.

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

$$\begin{aligned} z &= h_c + (h - h_c)\ cs) \\ c(s) &= \frac{[\tanh(\theta\,(s+b)) - \tanh(\theta\,b)]}{2\ \sinh(\theta)} \end{aligned} \tag{4.16}$$

where h_c is the thermocline depth and θ and b are the surface and bottom control parameters such that $0 \leqslant \theta \leqslant 20$, and $0 \leqslant b \leqslant 1$. b has been designed to allow surface and/or bottom increase of the vertical resolution (Fig. 4.7).

4.3.5 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.

4.3.6 level bathymetry and mask

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

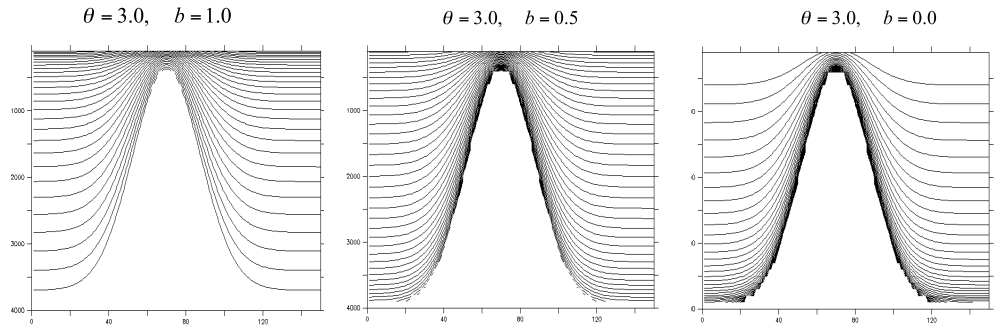

FIG. 4.7 – Examples of the stretching function applied to a sea mont ; from left to right : surface, surface and bottom, and bottom intensified resolutions

t-point. mbathy is computed from the meter bathymetry using the definiton of gdept as the number of t-points which gdept $\leq$ bathy.

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.

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

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

Note that *wmask* is not defined as it is exactly equal to *tmask* with the numerical indexing used (§ 4.1.3). Moreover, the specification of closed lateral boundaries requires that at least the first and last rows and columns of the *mbathy* array are set to zero. In the particular case of an east-west cyclical boundary condition, *mbathy* has its last column equal to the second one and its first column equal to the last but one (and so too the mask arrays) (see § 8.2).

5 Ocean Tracers (TRA)

Contents

Using the representation described in Chap. 4, several semi-discrete space forms of the tracer equations are available depending on the vertical coordinate used and on the physics used. In all the equations presented here, the masking has been omitted for simplicity. One must be aware that all the quantities are masked fields and that each time a mean or difference operator is used, the resulting field is multiplied by a mask.

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

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

NXT stands for next, referring to the time-stepping. From left to right, the terms on the rhs of the tracer equations are the advection (ADV), the lateral diffusion (LDF), the vertical diffusion (ZDF), the contributions from the external forcings (SBC : Surface Boundary Condition, QSR : penetrative Solar Radiation, and BBC : Bottom Boundary Condition), the contribution from the bottom boundary Layer (BBL) parametrisation, and an internal damping (DMP) term. The terms QSR, BBC, BBL and DMP are optional. The external forcings and parameterisations require complex inputs and complex calculations (e.g. bulk formulae, estimation of mixing coefficients) that are carried out in the SBC, LDF and ZDF modules and described in chapters §7, §9 and §10, 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 with associated modules *eosbn2.F90* and *phycst.F90*).

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

The user has the option of extracting each tendency term on the rhs of the tracer equation for output (**key_trdtra** is defined), as described in Chap. 13.

5.1 Tracer Advection (*traadv.F90*)

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

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

$$ADV_\tau = -\frac{1}{b_t}\left(\delta_i \left[e_{2u}\, e_{3u}\, u\, \tau_u \right] + \delta_j \left[e_{1v}\, e_{3v}\, v\, \tau_v \right] \right) - \frac{1}{e_{3t}}\, \delta_k \left[w\, \tau_w \right] \tag{5.1}$$

where τ is either T or S, and $b_t = e_{1t}\, e_{2t}\, e_{3t}$ is the volume of T-cells. The flux form in (5.1) implicitly requires the use of the continuity equation. Indeed, it is obtained by using the following equality : $\nabla \cdot (\mathbf{U}\,\mathrm{T}) = \mathbf{U} \cdot \nabla \mathrm{T}$ which results from the use of the continuity equation, $\nabla \cdot \mathbf{U} = 0$ or $\partial_t e_3 + e_3\ \nabla \cdot \mathbf{U} = 0$ in constant volume or variable volume case, respectively. Therefore it is of paramount importance to design the discrete analogue of the advection tendency so that it is consistent with the continuity equation in order to enforce the conservation properties of the continuous equations. In other words, by replacing τ by the number 1 in (5.1) we recover the discrete form of the continuity equation which is used to calculate the vertical velocity.

The key difference between the advection schemes available in *NEMO* is the choice made in space and time interpolation to define the value of the tracer at the velocity points (Fig. 5.1).

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

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 $\tau_w|_{k=1/2} = T_{k=1}$, *i.e.* the product of surface velocity (at $z = 0$) by the first level tracer value.

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

In all cases, this boundary condition retains local conservation of tracer. Global conservation is obtained in both rigid-lid and non-linear free surface cases, but not in the linear free surface case. Nevertheless, in the latter case, it is achieved to a good approximation

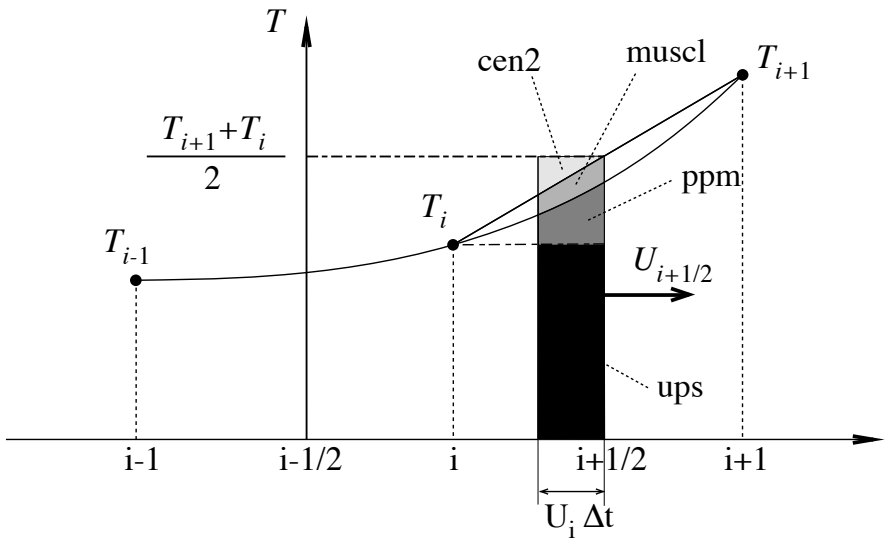

FIG. 5.1 – Schematic representation of some ways used to evaluate the tracer value at u-point and the amount of tracer exchanged between two neighbouring grid points. Upsteam biased scheme (ups) : the upstream value is used and the black area is exchanged. Piecewise parabolic method (ppm) : a parabolic interpolation is used and the black and dark grey areas are exchanged. Monotonic upstream scheme for conservative laws (muscl) : a parabolic interpolation is used and black, dark grey and grey areas are exchanged. Second order scheme (cen2) : the mean value is used and black, dark grey, grey and light grey areas are exchanged. Note that this illustration does not include the flux limiter used in ppm and muscl schemes.

since the non-conservative term is the product of the time derivative of the tracer and the free surface height, two quantities that are not correlated (see §2.2.2, and also **???**).

The velocity field that appears in (5.1) and (**??**) is the centred (*now*) *eulerian* ocean velocity (see Chap. 6). When eddy induced velocity (*eiv*) parameterisation is used it is the *now effective* velocity ($i.e.$ the sum of the eulerian and eiv velocities) which is used.

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

Note that (1) cen2, cen4 and TVD schemes require an explicit diffusion operator while the other schemes are diffusive enough so that they do not require additional diffusion ; (2) cen2, cen4, MUSCL2, and UBS are not *positive* schemes [1] , implying that false extrema are permitted. Their use is not recommended on passive tracers ; (3) It is recommended that the same advection-diffusion scheme is used on both active and passive tracers. Indeed, if a source or sink of a passive tracer depends on an active one, the difference of treatment of active and passive tracers can create very nice-looking frontal structures that are pure numerical artefacts. Nevertheless, most of our users set a different treatment on passive and active tracers, that's the reason why this possibility is offered. We strongly suggest them to perform a sensitivity experiment using a same treatment to assess the robustness of their results.

5.1.1 2^{nd} order centred scheme (cen2) (*ln_traadv_cen2*=true)

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

$$\tau_u^{cen2} = \overline{T}^{i+1/2} \tag{5.2}$$

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 (5.2) is the *now* tracer value. The centered second order advection is computed in the *traadv_cen2.F90* module. In this module, it is advantageous to combine the *cen2* scheme with an upstream scheme in specific areas which require a strong diffusion in order to avoid the generation of false extrema. These areas are the vicinity of large river mouths, some straits with coarse resolution, and the vicinity of ice cover area ($i.e.$ when the ocean temperature is close to the freezing point). This combined scheme has been included for specific grid points in the ORCA2 and ORCA4 configurations only. This is an obsolescent feature as the recommended advection scheme for the ORCA configuration is TVD (see §5.1.3).

Note that using the cen2 scheme, the overall tracer advection is of second order accuracy since both (5.1) and (5.2) have this order of accuracy.

5.1.2 4^{nd} order centred scheme (cen4) (*ln_traadv_cen4*=true)

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

$$\tau_u^{cen4} = \overline{T - \frac{1}{6}\,\delta_i \left[\delta_{i+1/2}[T] \right]}^{i+1/2} \tag{5.3}$$

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

Strictly speaking, the cen4 scheme is not a 4^{th} order advection scheme but a 4^{th} order evaluation of advective fluxes, since the divergence of advective fluxes (5.1) is kept at 2^{nd} order. The phrase "4^{th} order scheme" used in oceanographic literature is usually associated with the scheme presented here. Introducing a *true* 4^{th} order advection scheme is feasible but, for consistency reasons, it requires changes in the discretisation of the tracer advection together with changes in both the continuity equation and the momentum advection terms.

A direct consequence of the pseudo-fourth order nature of the scheme is that it is not non-diffusive, i.e. the global variance of a tracer is not preserved using *cen4*. Furthermore, it must be used in conjunction with an explicit diffusion operator to produce a sensible solution. The time-stepping is also performed using a leapfrog scheme in conjunction with an Asselin time-filter, so T in (5.3) is the *now* tracer.

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

5.1.3 Total Variance Dissipation scheme (TVD) (*ln_traadv_tvd*=true)

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

$$\tau_u^{ups} = \begin{cases} T_{i+1} & \text{if } u_{i+1/2} < 0 \\ T_i & \text{if } u_{i+1/2} \geq 0 \end{cases}$$
$$\tau_u^{tvd} = \tau_u^{ups} + c_u \left(\tau_u^{cen2} - \tau_u^{ups} \right) \tag{5.4}$$

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

For stability reasons (see §**??**), τ_u^{cen2} is evaluated in (5.4) using the *now* tracer while τ_u^{ups} is evaluated using the *before* tracer. In other words, the advective part of the scheme is time stepped with a leap-frog scheme while a forward scheme is used for the diffusive part.

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

The Monotone Upstream Scheme for Conservative Laws (MUSCL) has been implemented by **?**. In its formulation, the tracer at velocity points is evaluated assuming a linear tracer variation between two T-points (Fig.5.1). For example, in the i-direction :

$$\tau_u^{mus} = \begin{cases} \tau_i & +\frac{1}{2}\left(1 - \frac{u_{i+1/2}\,\Delta t}{e_{1u}}\right)\widetilde{\partial_i \tau} \quad \text{if } u_{i+1/2} \geqslant 0 \\ \tau_{i+1/2} & +\frac{1}{2}\left(1 + \frac{u_{i+1/2}\,\Delta t}{e_{1u}}\right)\widetilde{\partial_{i+1/2} \tau} \quad \text{if } u_{i+1/2} < 0 \end{cases} \tag{5.5}$$

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

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

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

5.1.5 Upstream-Biased Scheme (UBS) (*ln_traadv_ubs*=true)

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

$$\tau_u^{ubs} = \overline{T}^{i+1/2} - \frac{1}{6}\begin{cases} \tau"_i & \text{if } u_{i+1/2} \geqslant 0 \\ \tau"_{i+1} & \text{if } u_{i+1/2} < 0 \end{cases} \tag{5.6}$$

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

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

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

For stability reasons (see §**??**), the first term in (5.6) (which corresponds to a second order centred scheme) is evaluated using the *now* tracer (centred in time) while the second term (which is the diffusive part of the scheme), is evaluated using the *before* tracer (forward in time). This choice is discussed by **?** in the context of the QUICK advection scheme. UBS and QUICK schemes only differ by one coefficient. Replacing 1/6 with 1/8 in (5.6) leads to the QUICK advection scheme [**?**]. This option is not available through a namelist parameter, since the 1/6 coefficient is hard coded. Nevertheless it is quite easy to make the substitution in the *traadv_ubs.F90* module and obtain a QUICK scheme.

Four different options are possible for the vertical component used in the UBS scheme. τ_w^{ubs} can be evaluated using either *(a)* a centred 2^{nd} order scheme, or *(b)* a TVD scheme, or *(c)* an interpolation based on conservative parabolic splines following the **?** implementation of UBS in ROMS, or *(d)* a UBS. The 3^{rd} case has dispersion properties similar to an eighth-order accurate conventional scheme. The current reference version uses method b)

Note that :

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

(2) It is straightforward to rewrite (5.6) as follows :

$$\tau_u^{ubs} = \tau_u^{cen4} + \frac{1}{12} \begin{cases} +\tau"_i & \text{if } u_{i+1/2} \geqslant 0 \\ -\tau"_{i+1} & \text{if } u_{i+1/2} < 0 \end{cases} \tag{5.7}$$

or equivalently

$$u_{i+1/2}\, \tau_u^{ubs} = u_{i+1/2}\, \overline{T - \frac{1}{6}\, \delta_i \left[\delta_{i+1/2}[T] \right]}^{\,i+1/2} - \frac{1}{2} |u|_{i+1/2}\, \frac{1}{6}\, \delta_{i+1/2}[\tau"_i] \tag{5.8}$$

(5.7) has several advantages. Firstly, it clearly reveals that the UBS scheme is based on the fourth order scheme to which an upstream-biased diffusion term is added. Secondly, this emphasises that the 4^{th} order part (as well as the 2^{nd} order part as stated above) has to be evaluated at the *now* time step using (5.6). Thirdly, the diffusion term is in fact a biharmonic operator with an eddy coefficient which is simply proportional to the velocity : $A_u^{lm} = -\frac{1}{12}\, e_{1u}{}^3\, |u|$. Note that NEMO v3.3 still uses (5.6), not (5.7).

5.1.6 QUICKEST scheme (QCK) (*ln_traadv_qck*=true)

The Quadratic Upstream Interpolation for Convective Kinematics with Estimated Streaming Terms (QUICKEST) scheme proposed by **?** is the third order Godunov scheme. It is associated with the ULTIMATE QUICKEST limiter [**?**]. It has been implemented in NEMO by G. Reffray (MERCATOR-ocean) and can be found in the *traadv_qck.F90* module. The resulting scheme is quite expensive but *positive*. It can be used on both active

and passive tracers. However, the intrinsic diffusion of QCK makes its use risky in the vertical direction where the control of artificial diapycnal fluxes is of paramount importance. Therefore the vertical flux is evaluated using the CEN2 scheme. This no longer guarantees the positivity of the scheme. The use of TVD in the vertical direction (as for the UBS case) should be implemented to restore this property.

5.1.7 Piecewise Parabolic Method (PPM) (*ln_traadv_ppm*=true)

The Piecewise Parabolic Method (PPM) proposed by Colella and Woodward (1984) is based on a quadradic piecewise construction. Like the QCK scheme, it is associated with the ULTIMATE QUICKEST limiter [**?**]. It has been implemented in *NEMO* by G. Reffray (MERCATOR-ocean) but is not yet offered in the reference version 3.3.

5.2 Tracer Lateral Diffusion (*traldf.F90*)

```
!-----------------------------------------------------------------------
&namtra_ldf    !   lateral diffusion scheme for tracer
!-----------------------------------------------------------------------
   !                       !  Type of the operator :
   ln_traldf_lap    =  .true.   !  laplacian operator
   ln_traldf_bilap  =  .false.  !  bilaplacian operator
   !                       !  Direction of action  :
   ln_traldf_level  =  .false.  !  iso-level
   ln_traldf_hor    =  .false.  !  horizontal (geopotential)           (require "key_ldfslp" when ln_sco=T)
   ln_traldf_iso    =  .true.   !  iso-neutral                         (require "key_ldfslp")
   ln_traldf_grif   =  .false.  !  griffies skew flux formulation     (require "key_ldfslp")  ! UNDER TEST, DO NOT USE
   ln_traldf_gdia   =  .false.  !  griffies operator strfn diagnostics (require "key_ldfslp")  ! UNDER TEST, DO NOT USE
   !                       !  Coefficient
   rn_aht_0         =  2000.    !  horizontal eddy diffusivity for tracers [m2/s]
   rn_ahtb_0        =     0.    !  background eddy diffusivity for ldf_iso [m2/s]
   rn_aeiv_0        =  2000.    !  eddy induced velocity coefficient [m2/s]    (require "key_traldf_eiv")
/
```

The options available for lateral diffusion are a laplacian (rotated or not) or a biharmonic operator, the latter being more scale-selective (more diffusive at small scales). The specification of eddy diffusivity coefficients (either constant or variable in space and time) as well as the computation of the slope along which the operators act, are performed in the *ldftra.F90* and *ldfslp.F90* modules, respectively. This is described in Chap. 9. The lateral diffusion of tracers is evaluated using a forward scheme, $i.e.$ the tracers appearing in its expression are the *before* tracers in time, except for the pure vertical component that appears when a rotation tensor is used. This latter term is solved implicitly together with the vertical diffusion term (see §**??**).

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

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

$$D_T^{lT} = \frac{1}{b_t T} \left(\delta_i \left[A_u^{lT} \, \frac{e_{2u}\, e_{3u}}{e_{1u}} \, \delta_{i+1/2}[T] \right] + \delta_j \left[A_v^{lT} \, \frac{e_{1v}\, e_{3v}}{e_{2v}} \, \delta_{j+1/2}[T] \right] \right) \tag{5.9}$$

where b_t=$e_{1t}\, e_{2t}\, e_{3t}$ is the volume of T-cells. It is implemented in the *traadv_lap.F90* module.

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

Note that in the partial step z-coordinate (*ln_zps*=true), tracers in horizontally adjacent cells are located at different depths in the vicinity of the bottom. In this case, horizontal derivatives in (5.9) at the bottom level require a specific treatment. They are calculated in the *zpshde.F90* module, described in §5.9.

5.2.2 Rotated laplacian operator (iso) (*ln_traldf_lap*=true)

The general form of the second order lateral tracer subgrid scale physics (2.36) takes the following semi-discrete space form in z- and s-coordinates :

$$
\begin{aligned}
D_T^{lT} = \frac{1}{b_t} \Bigg\{ & \delta_i \left[A_u^{lT} \left(\frac{e_{2u}\, e_{3u}}{e_{1u}}\, \delta_{i+1/2}[T] - e_{2u}\, r_{1u}\, \overline{\overline{\delta_{k+1/2}[T]}}^{\,i+1/2,k} \right) \right] \\
& + \delta_j \left[A_v^{lT} \left(\frac{e_{1v}\, e_{3v}}{e_{2v}}\, \delta_{j+1/2}[T] - e_{1v}\, r_{2v}\, \overline{\overline{\delta_{k+1/2}[T]}}^{\,j+1/2,k} \right) \right] \\
& + \delta_k \Bigg[A_w^{lT} \Bigg(- e_{2w}\, r_{1w}\, \overline{\overline{\delta_{i+1/2}[T]}}^{\,i,k+1/2} \\
& \qquad - e_{1w}\, r_{2w}\, \overline{\overline{\delta_{j+1/2}[T]}}^{\,j,k+1/2} \\
& \qquad + \frac{e_{1w}\, e_{2w}}{e_{3w}} \left(r_{1w}^2 + r_{2w}^2 \right) \delta_{k+1/2}[T] \Bigg) \Bigg] \quad \Bigg\}
\end{aligned}
\tag{5.10}
$$

where b_t=$e_{1t}\, e_{2t}\, e_{3t}$ is the volume of T-cells, r_1 and r_2 are the slopes between the surface of computation (z- or s-surfaces) and the surface along which the diffusion operator acts ($i.e.$ horizontal or iso-neutral surfaces). It is thus used when, in addition to *ln_traldf_lap*= true, we have *ln_traldf_iso*=true, or both *ln_traldf_hor*=true and *ln_zco*=true. The way these slopes are evaluated is given in §9.2. At the surface, bottom and lateral boundaries, the turbulent fluxes of heat and salt are set to zero using the mask technique (see §8.1).

The operator in (5.10) involves both lateral and vertical derivatives. For numerical stability, the vertical second derivative must be solved using the same implicit time scheme as that used in the vertical physics (see §5.3). For computer efficiency reasons, this term is not computed in the *traldf_iso.F90* module, but in the *trazdf.F90* module where, if iso-neutral mixing is used, the vertical mixing coefficient is simply increased by $\frac{e_{1w}\, e_{2w}}{e_{3w}} \left(r_{1w}^2 + r_{2w}^2 \right)$.

This formulation conserves the tracer but does not ensure the decrease of the tracer variance. Nevertheless the treatment performed on the slopes (see §9) allows the model to run safely without any additional background horizontal diffusion [**?**]. An alternative scheme developed by **?** which preserves both tracer and its variance is also available in *NEMO* (*ln_traldf_grif*=true). A complete description of the algorithm is given in App.**??**.

Note that in the partial step z-coordinate (*ln_zps*=true), the horizontal derivatives at the bottom level in (5.10) require a specific treatment. They are calculated in module zpshde,

described in §5.9.

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

The lateral fourth order bilaplacian operator on tracers is obtained by applying (5.9) twice. The operator requires an additional assumption on boundary conditions : both first and third derivative terms normal to the coast are set to zero. It is used when, in addition to *ln_traldf_bilap*=true, we have *ln_traldf_level*=true, or both *ln_traldf_hor*=true and *ln_zco*=false. In both cases, it can contribute diapycnal mixing, although less than in the laplacian case. It is therefore not recommended.

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

5.2.4 Rotated bilaplacian operator (bilapg) (*ln_traldf_bilap*=true)

The lateral fourth order operator formulation on tracers is obtained by applying (5.10) twice. It requires an additional assumption on boundary conditions : first and third derivative terms normal to the coast, normal to the bottom and normal to the surface are set to zero. It can be found in the *traldf_bilapg.F90*.

It is used when, in addition to *ln_traldf_bilap*=true, we have *ln_traldf_iso*= .true., or both *ln_traldf_hor*=true and *ln_zco*=true. This rotated bilaplacian operator has never been seriously tested. There are no guarantees that it is either 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 with an unrotated operator.

5.3 Tracer Vertical Diffusion (*trazdf.F90*)

```
!-----------------------------------------------------------------------
&namzdf        !   vertical physics
!-----------------------------------------------------------------------
   rn_avm0     =   1.2e-4  !  vertical eddy viscosity   [m2/s]          (background Kz if not "key_zdfcst")
   rn_avt0     =   1.2e-5  !  vertical eddy diffusivity [m2/s]          (background Kz if not "key_zdfcst")
   nn_avb      =    0      !  profile for background avt & avm (=1) or not (=0)
   nn_havtb    =    0      !  horizontal shape for avtb (=1) or not (=0)
   ln_zdfevd   = .true.    !  enhanced vertical diffusion (evd) (T) or not (F)
   nn_evdm     =    0      !  evd apply on tracer (=0) or on tracer and momentum (=1)
   rn_avevd    =  100.     !  evd mixing coefficient [m2/s]
   ln_zdfnpc   = .false.   !  Non-Penetrative Convective algorithm (T) or not (F)
   nn_npc      =    1           !  frequency of application of npc
   nn_npcp     =  365           !  npc control print frequency
   ln_zdfexp   = .false.   !  time-stepping: split-explicit (T) or implicit (F) time stepping
   nn_zdfexp   =    3           !  number of sub-timestep for ln_zdfexp=T
/
```

The formulation of the vertical subgrid scale tracer physics is the same for all the vertical coordinates, and is based on a laplacian operator. The vertical diffusion operator

given by (2.36) takes the following semi-discrete space form :

$$\begin{aligned} D_T^{vT} &= \frac{1}{e_{3t}}\,\delta_k\left[\frac{A_w^{vT}}{e_{3w}}\delta_{k+1/2}[T]\right] \\ D_T^{vS} &= \frac{1}{e_{3t}}\,\delta_k\left[\frac{A_w^{vS}}{e_{3w}}\delta_{k+1/2}[S]\right] \end{aligned} \tag{5.11}$$

where A_w^{vT} and A_w^{vS} are the vertical eddy diffusivity coefficients on temperature and salinity, respectively. Generally, $A_w^{vT} = A_w^{vS}$ except when double diffusive mixing is parameterised ($i.e.$ **key_zdfddm** is defined). The way these coefficients are evaluated is given in §10 (ZDF). Furthermore, when iso-neutral mixing is used, both mixing coefficients are increased by $\frac{e_{1w}\,e_{2w}}{e_{3w}}\left(r_{1w}^2 + r_{2w}^2\right)$ to account for the vertical second derivative of (5.10).

At the surface and bottom boundaries, the turbulent fluxes of heat and salt must be specified. At the surface they are prescribed from the surface forcing and added in a dedicated routine (see §5.4.1), whilst at the bottom they are set to zero for heat and salt unless a geothermal flux forcing is prescribed as a bottom boundary condition (see §5.4.3).

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

5.4 External Forcing

5.4.1 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.

Due to interactions and mass exchange of water (F_{mass}) with other Earth system components ($i.e.$ atmosphere, sea-ice, land), the change in the heat and salt content of the surface layer of the ocean is due both to the heat and salt fluxes crossing the sea surface (not linked with F_{mass}) and to the heat and salt content of the mass exchange.

The surface module (*sbcmod.F90*, see §7) provides the following forcing fields (used on tracers) :

• Q_{ns}, the non-solar part of the net surface heat flux that crosses the sea surface (i.e. the difference between the total surface heat flux and the fraction of the short wave flux that penetrates into the water column, see §5.4.2)

• *emp*, the mass flux exchanged with the atmosphere (evaporation minus precipitation)

• emp_S, an equivalent mass flux taking into account the effect of ice-ocean mass exchange

• *rnf*, the mass flux associated with runoff (see §7.8 for further detail of how it acts on temperature and salinity tendencies)

The emp_S field is not simply the budget of evaporation-precipitation+freezing-melting because the sea-ice is not currently embedded in the ocean but levitates above it. There is no mass exchanged between the sea-ice and the ocean. Instead we only take into account the salt flux associated with the non-zero salinity of sea-ice, and the concentration/dilution effect due to the freezing/melting (F/M) process. These two parts of the forcing are then converted into an equivalent mass flux given by $emp_S - emp$. As a result of this mess, the surface boundary condition on temperature and salinity is applied as follows :

In the nonlinear free surface case (**key_vvl** is defined) :

$$\begin{aligned} F^T &= \frac{1}{\rho_o \; C_p \;\; e_{3t}|_{k=1}} \quad \overline{(Q_{ns} - emp \; C_p \;\; T|_{k=1})}^t \\ F^S &= \frac{1}{\rho_o \;\; e_{3t}|_{k=1}} \qquad \overline{((emp_S - emp) \;\; S|_{k=1})}^t \end{aligned} \tag{5.12}$$

In the linear free surface case (**key_vvl** not defined) :

$$\begin{aligned} F^T &= \frac{1}{\rho_o \; C_p \;\; e_{3t}|_{k=1}} \qquad \overline{Q_{ns}}^t \\ F^S &= \frac{1}{\rho_o \;\; e_{3t}|_{k=1}} \qquad \overline{(emp_S \;\; S|_{k=1})}^t \end{aligned} \tag{5.13}$$

where $\overline{x}^t$ means that x is averaged over two consecutive time steps ($t-\Delta t/2$ and $t+\Delta t/2$). Such time averaging prevents the divergence of odd and even time step (see §3).

The two set of equations, (5.12) and (5.13), are obtained by assuming that the temperature of precipitation and evaporation are equal to the ocean surface temperature and that their salinity is zero. Therefore, the heat content of the *emp* budget must be added to the temperature equation in the variable volume case, while it does not appear in the constant volume case. Similarly, the *emp* budget affects the ocean surface salinity in the constant volume case (through the concentration dilution effect) while it does not appears explicitly in the variable volume case since salinity change will be induced by volume change. In both constant and variable volume cases, surface salinity will change with ice-ocean salt flux and F/M flux (both contained in $emp_S - emp$) without mass exchanges.

Note that the concentration/dilution effect due to F/M is computed using a constant ice salinity as well as a constant ocean salinity. This approximation suppresses the correlation between *SSS* and F/M flux, allowing the ice-ocean salt exchanges to be conservative. Indeed, if this approximation is not made, even if the F/M budget is zero on average over the whole ocean domain and over the seasonal cycle, the associated salt flux is not zero, since sea-surface salinity and F/M flux are intrinsically correlated (high *SSS* are found where freezing is strong whilst low *SSS* is usually associated with high melting areas).

Even using this approximation, an exact conservation of heat and salt content is only achieved in the variable volume case. In the constant volume case, there is a small imbalance associated with the product $(\partial_t \eta - emp) * SSS$. Nevertheless, the salt content

variation is quite small and will not induce a long term drift as there is no physical reason for $(\partial_t \eta - emp)$ and *SSS* to be correlated [**?**]. Note that, while quite small, the imbalance in the constant volume case is larger than the imbalance associated with the Asselin time filter [**?**]. This is the reason why the modified filter is not applied in the constant volume case.

5.4.2 Solar Radiation Penetration (*traqsr.F90*)

```
!-----------------------------------------------------------------------
&namtra_qsr    !   penetrative solar radiation
!-----------------------------------------------------------------------
!             !  file name  ! frequency (hours) ! variable  ! time interp. !  clim  ! 'yearly'/ ! weights  ! rotation !
!             !             !  (if <0  months)  !   name    !   (logical)  !  (T/F) ! 'monthly' ! filename ! pairing  !
   sn_chl      ='chlorophyll',        -1         , 'CHLA'    ,   .true.     , .true. , 'yearly'  , ''       , ''

   cn_dir      = './'      !  root directory for the location of the runoff files
   ln_traqsr   = .true.    !  Light penetration (T) or not (F)
   ln_qsr_rgb  = .true.    !  RGB (Red-Green-Blue) light penetration
   ln_qsr_2bd  = .false.   !  2 bands              light penetration
   ln_qsr_bio  = .false.   !  bio-model light penetration
   nn_chldta   =      1    !  RGB : Chl data (=1) or cst value (=0)
   rn_abs      =   0.58    !  RGB & 2 bands: fraction of light (rn_si1)
   rn_si0      =   0.35    !  RGB & 2 bands: shortess depth of extinction
   rn_si1      =   23.0    !  2 bands: longest depth of extinction
/
```

When the penetrative solar radiation option is used (*ln_flxqsr*=true), the solar radiation penetrates the top few tens of meters of the ocean. If it is not used (*ln_flxqsr*=false) all the heat flux is absorbed in the first ocean level. Thus, in the former case a term is added to the time evolution equation of temperature (2.1d) and the surface boundary condition is modified to take into account only the non-penetrative part of the surface heat flux :

$$\begin{aligned} \frac{\partial T}{\partial t} &= \ldots + \frac{1}{\rho_o \, C_p \, e_3} \, \frac{\partial I}{\partial k} \\ Q_{ns} &= Q_{\text{Total}} - Q_{sr} \end{aligned} \tag{5.14}$$

where Q_{sr} is the penetrative part of the surface heat flux ($i.e.$ the shortwave radiation) and I is the downward irradiance ($I|_{z=\eta} = Q_{sr}$). The additional term in (5.14) is discretized as follows :

$$\frac{1}{\rho_o \, C_p \, e_3} \, \frac{\partial I}{\partial k} \equiv \frac{1}{\rho_o \, C_p \, e_{3t}} \delta_k \left[I_w \right] \tag{5.15}$$

The shortwave radiation, Q_{sr}, consists of energy distributed across a wide spectral range. The ocean is strongly absorbing for wavelengths longer than 700 nm and these wavelengths contribute to heating the upper few tens of centimetres. The fraction of Q_{sr} that resides in these almost non-penetrative wavebands, R, is $\sim 58\%$ (specified through namelist parameter *rn_abs*). It is assumed to penetrate the ocean with a decreasing exponential profile, with an e-folding depth scale, ξ_0, of a few tens of centimetres (typically $\xi_0 = 0.35 \; m$ set as *rn_si0* in the namtra_qsr namelist). For shorter wavelengths (400-700 nm), the ocean is more transparent, and solar energy propagates to larger depths where it contributes to local heating. The way this second part of the solar energy penetrates into the ocean depends on which formulation is chosen. In the simple 2-waveband light penetration scheme (*ln_qsr_2bd*=true) a chlorophyll-independent monochromatic formulation

is chosen for the shorter wavelengths, leading to the following expression [**?**] :

$$I(z) = Q_{sr} \left[R e^{-z/\xi_0} + (1 - R)\, e^{-z/\xi_1} \right] \tag{5.16}$$

where ξ_1 is the second extinction length scale associated with the shorter wavelengths. It is usually chosen to be 23 m by setting the *rn_si0* namelist parameter. The set of default values (ξ_0, ξ_1, R) corresponds to a Type I water in Jerlov's (1968) classification (oligotrophic waters).

Such assumptions have been shown to provide a very crude and simplistic representation of observed light penetration profiles (**?**, see also Fig.5.2). Light absorption in the ocean depends on particle concentration and is spectrally selective. **?** has shown that an accurate representation of light penetration can be provided by a 61 waveband formulation. Unfortunately, such a model is very computationally expensive. Thus, **?** have constructed a simplified version of this formulation in which visible light is split into three wavebands : blue (400-500 nm), green (500-600 nm) and red (600-700nm). For each wave-band, the chlorophyll-dependent attenuation coefficient is fitted to the coefficients computed from the full spectral model of **?** (as modified by **?**), assuming the same power-law relationship. As shown in Fig.5.2, this formulation, called RGB (Red-Green-Blue), reproduces quite closely the light penetration profiles predicted by the full spectal model, but with much greater computational efficiency. The 2-bands formulation does not reproduce the full model very well.

The RGB formulation is used when *ln_qsr_rgb*=true. The RGB attenuation coefficients ($i.e.$ the inverses of the extinction length scales) are tabulated over 61 nonuniform chlorophyll classes ranging from 0.01 to 10 g.Chl/L (see the routine *trc_oce_rgb* in *trc_oce.F90* module). Three types of chlorophyll can be chosen in the RGB formulation : (1) a constant 0.05 g.Chl/L value everywhere (*nn_chdta*=0) ; (2) an observed time varying chlorophyll (*nn_chdta*=1) ; (3) simulated time varying chlorophyll by TOP biogeochemical model (*ln_qsr_bio*=true). In the latter case, the RGB formulation is used to calculate both the phytoplankton light limitation in PISCES or LOBSTER and the oceanic heating rate.

The trend in (5.15) associated with the penetration of the solar radiation is added to the temperature trend, and the surface heat flux is modified in routine *traqsr.F90*.

When the z-coordinate is preferred to the s-coordinate, the depth of $w-$levels does not significantly vary with location. The level at which the light has been totally absorbed ($i.e.$ it is less than the computer precision) is computed once, and the trend associated with the penetration of the solar radiation is only added down to that level. Finally, note that when the ocean is shallow (< 200 m), part of the solar radiation can reach the ocean floor. In this case, we have chosen that all remaining radiation is absorbed in the last ocean level ($i.e.$ I is masked).

5.4.3 Bottom Boundary Condition (*trabbc.F90*)

```
!-----------------------------------------------------------------------
&nambbc        !   bottom temperature boundary condition
!-----------------------------------------------------------------------
```

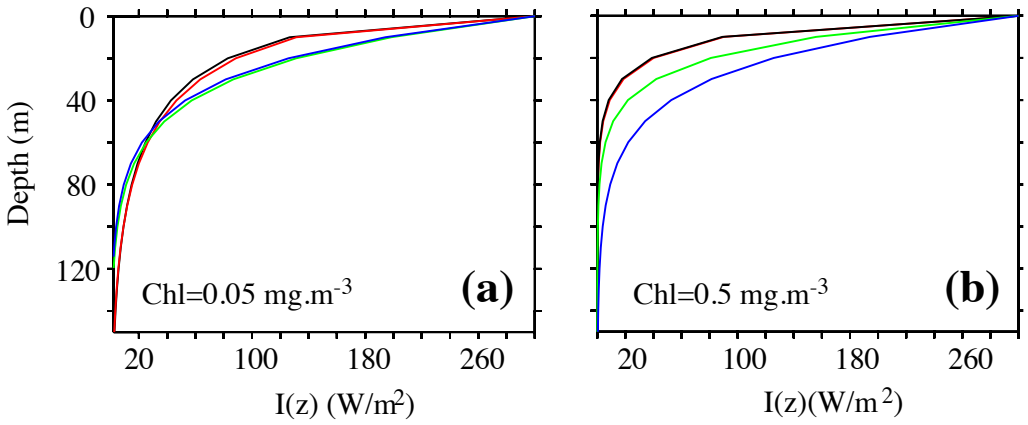

FIG. 5.2 – Penetration profile of the downward solar irradiance calculated by four models. Two waveband chlorophyll-independent formulation (blue), a chlorophyll-dependent monochromatic formulation (green), 4 waveband RGB formulation (red), 61 waveband Morel (1988) formulation (black) for a chlorophyll concentration of (a) Chl=0.05 mg/m^3 and (b) Chl=0.5 mg/m^3. From **?**.

```
   ln_trabbc    = .true.     !  Apply a geothermal heating at the ocean bottom
   nn_geoflx    =    2       !  geothermal heat flux: = 0 no flux
                             !     = 1 constant flux
                             !     = 2 variable flux (read in geothermal_heating.nc in mW/m2)
   rn_geoflx_cst = 86.4e-3 !  Constant value of geothermal heat flux [W/m2]
/
```

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

The presence of geothermal heating is controlled by setting the namelist parameter *ln_trabbc* to true. Then, when *nn_geoflx* is set to 1, a constant geothermal heating is introduced whose value is given by the *nn_geoflx_cst*, which is also a namelist parameter. When *nn_geoflx* is set to 2, a spatially varying geothermal heat flux is introduced which is provided in the *geothermal_heating.nc* NetCDF file (Fig.5.3) [**?**].

5.5 Bottom Boundary Layer (*trabbl.F90* - key_trabbl)

```
!-----------------------------------------------------------------------
&nambbl        !   bottom boundary layer scheme
```

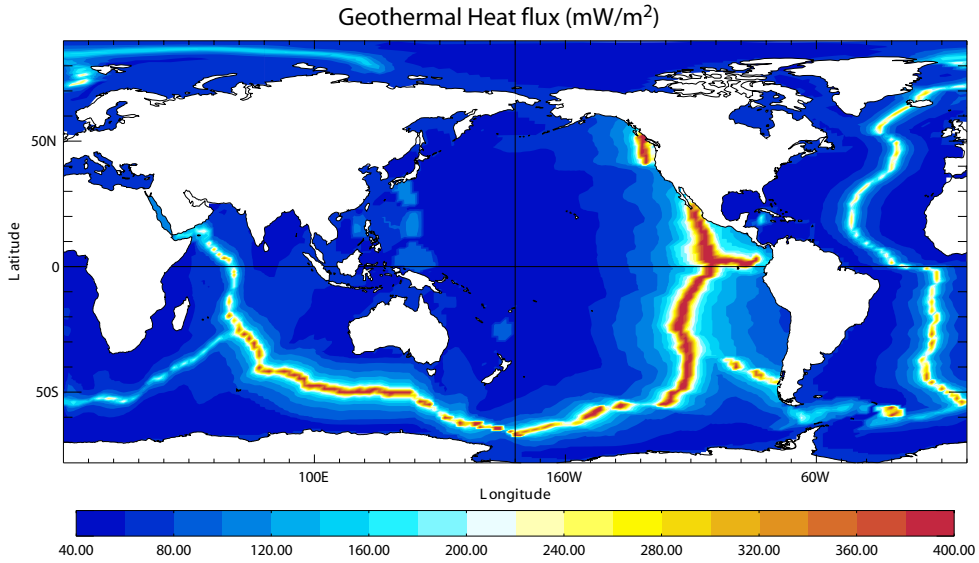

FIG. 5.3 – Geothermal Heat flux (in $mW.m^{-2}$) used by **?**. It is inferred from the age of the sea floor and the formulae of **?**.

```
!-----------------------------------------------------------------------
   nn_bbl_ldf  =  1      !  diffusive bbl (=1)   or not (=0)
   nn_bbl_adv  =  0      !  advective bbl (=1/2) or not (=0)
   rn_ahtbbl   =  1000.  !  lateral mixing coefficient in the bbl  [m2/s]
   rn_gambbl   =  10.    !  advective bbl coefficient                [s]
/
```

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

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

Furthermore, it only connects ocean bottom cells, and therefore does not include all the improvements introduced by **?**.

5.5.1 Diffusive Bottom Boundary layer (*nn_bbl_ldf*=1)

When applying sigma-diffusion (**key_trabbl** defined and *nn_bbl_ldf* set to 1), the diffusive flux between two adjacent cells at the ocean floor is given by

$$\mathbf{F}_\sigma = A_l^\sigma \, \nabla_\sigma T \tag{5.17}$$

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

$$A_l^\sigma(i,j,t) = \begin{cases} A_{bbl} & \text{if} \quad \nabla_\sigma \rho \cdot \nabla H < 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.18}$$

where A_{bbl} is the BBL diffusivity coefficient, given by the namelist parameter *rn_ahtbbl* and usually set to a value much larger than the one used for lateral mixing in the open ocean. The constraint in (5.18) implies that sigma-like diffusion only occurs when the density above the sea floor, at the top of the slope, is larger than in the deeper ocean (see green arrow in Fig.5.4). In practice, this constraint is applied separately in the two horizontal directions, and the density gradient in (5.18) is evaluated with the log gradient formulation :

$$\nabla_\sigma \rho / \rho = \alpha \, \nabla_\sigma T + \beta \, \nabla_\sigma S \tag{5.19}$$

where ρ, α and β are functions of $\overline{T}^\sigma$, $\overline{S}^\sigma$ and $\overline{H}^\sigma$, the along bottom mean temperature, salinity and depth, respectively.

5.5.2 Advective Bottom Boundary Layer (*nn_bbl_adv*= 1 or 2)

When applying an advective BBL (*nn_bbl_adv* = 1 or 2), an overturning circulation is added which connects two adjacent bottom grid-points only if dense water overlies less dense water on the slope. The density difference causes dense water to move down the slope.

nn_bbl_adv = 1 : the downslope velocity is chosen to be the Eulerian ocean velocity just above the topographic step (see black arrow in Fig.5.4) [**?**]. It is a *conditional advection*, that is, advection is allowed only if dense water overlies less dense water on the slope ($i.e.$ $\nabla_\sigma \rho \cdot \nabla H < 0$) and if the velocity is directed towards greater depth ($i.e.$ $\mathbf{U} \cdot \nabla \mathrm{H} > 0$).

nn_bbl_adv = 2 : the downslope velocity is chosen to be proportional to $\Delta\rho$, the density difference between the higher cell and lower cell densities [**?**]. The advection is allowed only if dense water overlies less dense water on the slope ($i.e.$ $\nabla_\sigma \rho \cdot \nabla H < 0$). For

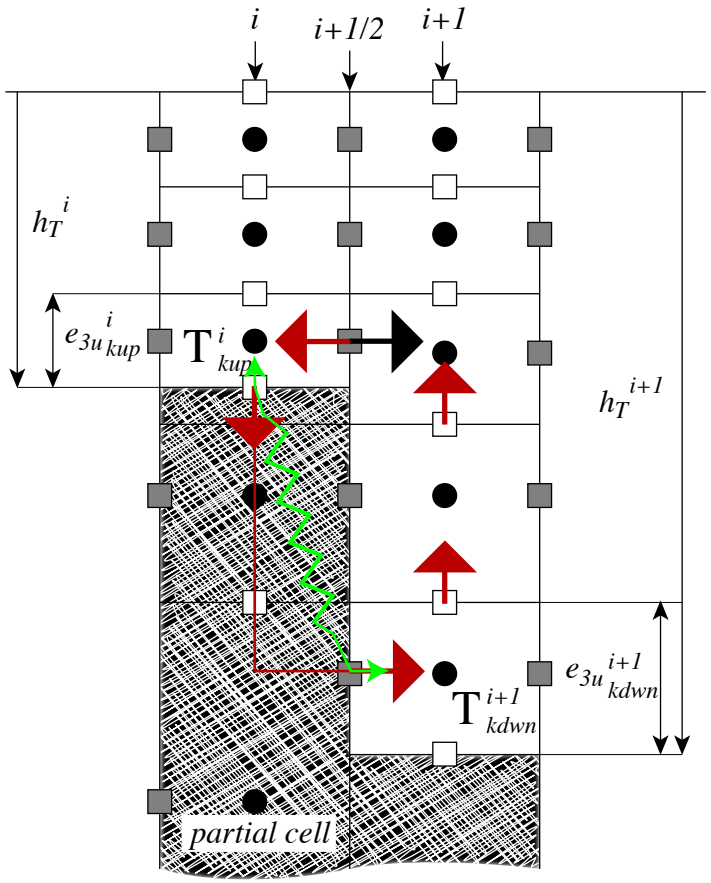

FIG. 5.4 – Advective/diffusive Bottom Boundary Layer. The BBL parameterisation is activated when ρ^i_{kup} is larger than ρ^{i+1}_{kdnw}. Red arrows indicate the additional overturning circulation due to the advective BBL. The transport of the downslope flow is defined either as the transport of the bottom ocean cell (black arrow), or as a function of the along slope density gradient. The green arrow indicates the diffusive BBL flux directly connecting kup and $kdwn$ ocean bottom cells. connection

example, the resulting transport of the downslope flow, here in the i-direction (Fig.5.4), is simply given by the following expression :

$$u^{tr}_{bbl} = \gamma\, g \frac{\Delta\rho}{\rho_o} e_{1u}\; min\left(e_{3u\,kup}, e_{3u\,kdwn}\right) \tag{5.20}$$

where γ, expressed in seconds, is the coefficient of proportionality provided as *rn_gambbl*, a namelist parameter, and *kup* and *kdwn* are the vertical index of the higher and lower cells, respectively. The parameter γ should take a different value for each bathymetric step, but

for simplicity, and because no direct estimation of this parameter is available, a uniform value has been assumed. The possible values for γ range between 1 and 10 s [**?**].

Scalar properties are advected by this additional transport $(u^{tr}_{bbl}, v^{tr}_{bbl})$ using the upwind scheme. Such a diffusive advective scheme has been chosen to mimic the entrainment between the downslope plume and the surrounding water at intermediate depths. The entrainment is replaced by the vertical mixing implicit in the advection scheme. Let us consider as an example the case displayed in Fig.5.4 where the density at level (i, kup) is larger than the one at level $(i, kdwn)$. The advective BBL scheme modifies the tracer time tendency of the ocean cells near the topographic step by the downslope flow (5.21), the horizontal (5.22) and the upward (5.23) return flows as follows :

$$\partial_t T^{do}_{kdw} \equiv \partial_t T^{do}_{kdw} + \frac{u^{tr}_{bbl}}{b^{do}_{t\,kdw}} \left(T^{sh}_{kup} - T^{do}_{kdw} \right) \tag{5.21}$$

$$\partial_t T^{sh}_{kup} \equiv \partial_t T^{sh}_{kup} + \frac{u^{tr}_{bbl}}{b^{sh}_{t\,kup}} \left(T^{do}_{kup} - T^{sh}_{kup} \right) \tag{5.22}$$

and for $k = kdw - 1,\ ...,\ kup$:

$$\partial_t T^{do}_{k} \equiv \partial_t S^{do}_{k} + \frac{u^{tr}_{bbl}}{b^{do}_{t\,k}} \left(T^{do}_{k+1} - T^{sh}_{k} \right) \tag{5.23}$$

where b_t is the T-cell volume.

Note that the BBL transport, $(u^{tr}_{bbl}, v^{tr}_{bbl})$, is available in the model outputs. It has to be used to compute the effective velocity as well as the effective overturning circulation.

5.6 Tracer damping (*tradmp.F90*)

```
!-----------------------------------------------------------------------
&namtra_dmp    !   tracer: T & S newtonian damping                    ('key_tradmp')
!-----------------------------------------------------------------------
   nn_hdmp     =   -1      !  horizontal shape =-1, damping in Med and Red Seas only
                           !                   =XX, damping poleward of XX degrees (XX>0)
                           !                      + F(distance-to-coast) + Red and Med Seas
   nn_zdmp     =    1      !  vertical   shape =0    damping throughout the water column
                           !                   =1 no damping in the mixing layer (kz  criteria)
                           !                   =2 no damping in the mixed  layer (rho crieria)
   rn_surf     =   50.     !  surface time scale of damping   [days]
   rn_bot      =  360.     !  bottom  time scale of damping   [days]
   rn_dep      =  800.     !  depth of transition between rn_surf and rn_bot [meters]
   nn_file     =    0      !  create a damping.coeff NetCDF file (=1) or not (=0)
/

!-----------------------------------------------------------------------
&namdta_tem    !   data : temperature                                 ("key_dtatem")
!-----------------------------------------------------------------------
!              ! file name ! frequency (hours)    ! variable ! time interp. ! clim  !'yearly' or ! weights  ! rotation !
!              !           !  (if <0  months)     !   name   !  (logical)   ! (T/F) ! 'monthly'  ! filename ! pairing  !
   sn_tem = 'data_1m_potential_temperature_nomask', -1,'votemper',  .true.  , .true., 'yearly'   , ' '      , ' '
   !
   cn_dir       = './'     !  root directory for the location of the runoff files
/

!-----------------------------------------------------------------------
&namdta_sal    !   data : salinity                                    ("key_dtasal")
!-----------------------------------------------------------------------
!              ! file name ! frequency (hours)    ! variable ! time interp. ! clim  !'yearly' or ! weights  ! rotation !
!              !           !  (if <0  months)     !   name   !  (logical)   ! (T/F) ! 'monthly'  ! filename ! pairing  !
   sn_sal      =  'data_1m_salinity_nomask',  -1  ,'vosaline',    .true.    , .true., 'yearly'   , ''       , ' '
   !
   cn_dir      = './'     !  root directory for the location of the runoff files
/
```

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

$$\begin{aligned} \frac{\partial T}{\partial t} &= \cdots - \gamma \, (T - T_o) \\ \frac{\partial S}{\partial t} &= \cdots - \gamma \, (S - S_o) \end{aligned} \tag{5.24}$$

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_dtatem** and **key_dtasal** are defined and fill in *namdta_tem* and *namdta_sal namelists* ($i.e.$ that T_o and S_o are read using *fldread.F90*, see §7.2.1). The restoring coefficient γ is a three-dimensional array initialized by the user in routine *dtacof* also located in module *tradmp.F90*.

The two main cases in which (5.24) 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, γ takes large values (equivalent to a time scale of a few days) whereas it is zero in the interior of the model domain. The second case corresponds to the use of the robust diagnostic method [**?**]. It allows us to find the velocity field consistent with the model dynamics whilst having a T, S field close to a given climatological field (T_o, S_o). The time scale associated with S_o is generally not a constant but spatially varying in order to respect other properties. For example, it is usually set to zero in the mixed layer (defined either on a density or S_o criterion) [**?**] and in the equatorial region [**???**] since these two regions have a short time scale of adjustment ; while smaller γ are used in the deep ocean where the typical time scale is long [**?**]. In addition the time scale is reduced (even to zero) along the western boundary to allow the model to reconstruct its own western boundary structure in equilibrium with its physics. The choice of the shape of the Newtonian damping is controlled by two namelist parameters *nn_hdmp* and *nn_zdmp*. The former allows us to specify : the width of the equatorial band in which no damping is applied ; a decrease in the vicinity of the coast ; and a damping everywhere in the Red and Med Seas. The latter sets whether damping should act in the mixed layer or not. The time scale associated with the damping depends on the depth as a hyperbolic tangent, with *rn_surf* as surface value, *rn_bot* as bottom value and a transition depth of *rn_dep*.

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

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

5.7 Tracer time evolution (*tranxt.F90*)

```
!-----------------------------------------------------------------------
&namdom        !   space and time domain (bathymetry, mesh, timestep)
!-----------------------------------------------------------------------
   nn_bathy    =    1      !  compute (=0) or read (=1) the bathymetry file
   nn_closea    =   0      !  remove (=0) or keep (=1) closed seas and lakes (ORCA)
   nn_msh      =    0      !  create (=1) a mesh file or not (=0)
   rn_hmin     =   -3.     !  min depth of the ocean (>0) or min number of ocean level (<0)
   rn_e3zps_min=   20.     !  partial step thickness is set larger than the minimum of
   rn_e3zps_rat=    0.1    !  rn_e3zps_min and rn_e3zps_rat*e3t, with 0<rn_e3zps_rat<1
                           !
   rn_rdt      = 5760.     !  time step for the dynamics (and tracer if nn_acc=0)
   nn_baro     =   64      !  number of barotropic time step            ("key_dynspg_ts")
   rn_atfp     =    0.1    !  asselin time filter parameter
   nn_acc      =    0      !  acceleration of convergence : =1      used, rdt < rdttra(k)
                           !                                =0, not used, rdt = rdttra
   rn_rdtmin   = 28800.    !  minimum time step on tracers (used if nn_acc=1)
   rn_rdtmax   = 28800.    !  maximum time step on tracers (used if nn_acc=1)
   rn_rdth     =  800.     !  depth variation of tracer time step  (used if nn_acc=1)
/
```

The general framework for tracer time stepping is a modified leap-frog scheme [**?**], $i.e.$ a three level centred time scheme associated with a Asselin time filter (cf. §3.5) :

$$
\begin{aligned}
(e_{3t}T)^{t+\Delta t} &= (e_{3t}T)_f^{t-\Delta t} && +2\,\Delta t\, e_{3t}^t\,\mathrm{RHS}^t \\
(e_{3t}T)_f^t &= (e_{3t}T)^t && +\gamma\left[(e_{3t}T)_f^{t-\Delta t}-2(e_{3t}T)^t+(e_{3t}T)^{t+\Delta t}\right] \\
& && -\gamma\,\Delta t\left[Q^{t+\Delta t/2}-Q^{t-\Delta t/2}\right]
\end{aligned}
\tag{5.25}
$$

where RHS is the right hand side of the temperature equation, the subscript f denotes filtered values, γ is the Asselin coefficient, and S is the total forcing applied on T ($i.e.$ fluxes plus content in mass exchanges). γ is initialized as *rn_atfp* (**namelist** parameter). Its default value is *rn_atfp*=10^{-3}. Note that the forcing correction term in the filter is not applied in linear free surface (*lk_vvl*=false) (see §5.4.1. Not also that in constant volume case, the time stepping is performed on T, not on its content, $e_{3t}T$.

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

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

5.8 Equation of State (*eosbn2.F90*)

```
!-----------------------------------------------------------------------
&nameos        !   ocean physical parameters
!-----------------------------------------------------------------------
   nn_eos      =   0       !  type of equation of state and Brunt-Vaisala frequency
                           !     = 0, UNESCO (formulation of Jackett and McDougall (1994) and of McDougall (1987) )
                           !     = 1, linear: rho(T)   = rau0 * ( 1.028 - ralpha * T )
                           !     = 2, linear: rho(T,S) = rau0 * ( rbeta * S - ralpha * T )
   rn_alpha    =   2.0e-4  !  thermal expension coefficient (nn_eos= 1 or 2)
   rn_beta     =   7.7e-4  !  saline  expension coefficient (nn_eos= 2)
/
```

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

In the computer code, a *true* density anomaly, $d_a = \rho/\rho_o - 1$, is computed, with ρ_o a reference volumic mass. Called *rau0* in the code, ρ_o is defined in *phycst.F90*, and a value of $1,035\ Kg/m^3$. This is a sensible choice for the reference density used in a Boussinesq ocean climate model, as, with the exception of only a small percentage of the ocean, density in the World Ocean varies by no more than 2% from $1,035\ kg/m^3$ [**?**].

The default option (namelist parameter *nn_eos*=0) is the **?** equation of state. Its use is highly recommended. However, for process studies, it is often convenient to use a linear approximation of the density. With such an equation of state there is no longer a distinction between *in situ* and *potential* density and both cabbeling and thermobaric effects are removed. Two linear formulations are available : a function of T only (*nn_eos*=1) and a function of both T and S (*nn_eos*=2) :

$$\begin{aligned} d_a(T) &= \rho(T)/\rho_o - 1 = \ 0.0285 - \alpha\ T \\ d_a(T,S) &= \rho(T,S)/\rho_o - 1 = \ \beta\ S - \alpha\ T \end{aligned} \tag{5.26}$$

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

5.8.2 Brunt-Vaisälä Frequency (*nn_eos* = 0, 1 or 2)

An accurate computation of the ocean stability (i.e. of N, the brunt-Vaisälä frequency) is of paramount importance as it is used in several ocean parameterisations (namely TKE,

KPP, Richardson number dependent vertical diffusion, enhanced vertical diffusion, non-penetrative convection, iso-neutral diffusion). In particular, one must be aware that N^2 has to be computed with an *in situ* reference. The expression for N^2 depends on the type of equation of state used (*nn_eos* namelist parameter).

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

$$N^2 = \frac{g}{e_{3w}} \beta \left(\alpha/\beta \, \delta_{k+1/2}[T] - \delta_{k+1/2}[S] \right) \tag{5.27}$$

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

When a linear equation of state is used (*nn_eos*=1 or 2, (5.27) reduces to :

$$N^2 = \frac{g}{e_{3w}} \left(\beta \, \delta_{k+1/2}[S] - \alpha \, \delta_{k+1/2}[T] \right) \tag{5.28}$$

where α and β are the constant coefficients used to defined the linear equation of state (5.26).

5.8.3 Specific Heat (*phycst.F90*)

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

5.8.4 Freezing Point of Seawater

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

$$\begin{aligned} T_f(S,p) = \Big(-0.0575 + 1.710523\ 10^{-3} \sqrt{S} - 2.154996\ 10^{-4}\ S \Big)\ S \\ -7.53\,10^{-3}\ p \end{aligned} \tag{5.29}$$

(5.29) 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 (5.29) (last term) have been dropped. The freezing point is computed through *tfreez*, a FORTRAN function that can be found in *eosbn2.F90*.

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

With partial bottom cells (*ln_zps*=true), in general, tracers in horizontally adjacent cells live at different depths. Horizontal gradients of tracers are needed for horizontal diffusion (*traldf.F90* module) and for the hydrostatic pressure gradient (*dynhpg.F90* module) to be active. Before taking horizontal gradients between the tracers next to the bottom, a linear interpolation in the vertical is used to approximate the deeper tracer as if it actually lived at the depth of the shallower tracer point (Fig. 5.5). For example, for temperature in the i-direction the needed interpolated temperature, $\widetilde{T}$, is :

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

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

$$\begin{aligned} \delta_{i+1/2} T &= \begin{cases} \widetilde{T} - T^{i} & \text{if } e_{3w}^{i+1} \geq e_{3w}^{i} \\ \\ T^{i+1} - \widetilde{T} & \text{if } e_{3w}^{i+1} < e_{3w}^{i} \end{cases} \\ \\ \overline{T}^{i+1/2} &= \begin{cases} (\widetilde{T} - T^{i})/2 & \text{if } e_{3w}^{i+1} \geq e_{3w}^{i} \\ \\ (T^{i+1} - \widetilde{T})/2 & \text{if } e_{3w}^{i+1} < e_{3w}^{i} \end{cases} \end{aligned} \tag{5.30}$$

The computation of horizontal derivative of tracers as well as of density is performed once for all at each time step in *zpshde.F90* module and stored in shared arrays to be used when needed. It has to be emphasized that the procedure used to compute the interpolated density, $\widetilde{\rho}$, is not the same as that used for T and S. Instead of forming a linear approximation of density, we compute $\widetilde{\rho}$ from the interpolated values of T and S, and the pressure at a u-point (in the equation of state pressure is approximated by depth, see §5.8.1) :

$$\widetilde{\rho} = \rho(\widetilde{T}, \widetilde{S}, z_u) \quad \text{where} \quad z_u = \min\left(z_T^{i+1}, z_T^{i}\right) \tag{5.31}$$

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 (§6.4) and the slopes of neutral surfaces (§9.2)

Note that in almost all the advection schemes presented in this Chapter, both averaging and differencing operators appear. Yet (5.30) has not been used in these schemes : in contrast to diffusion and pressure gradient computations, no correction for partial steps

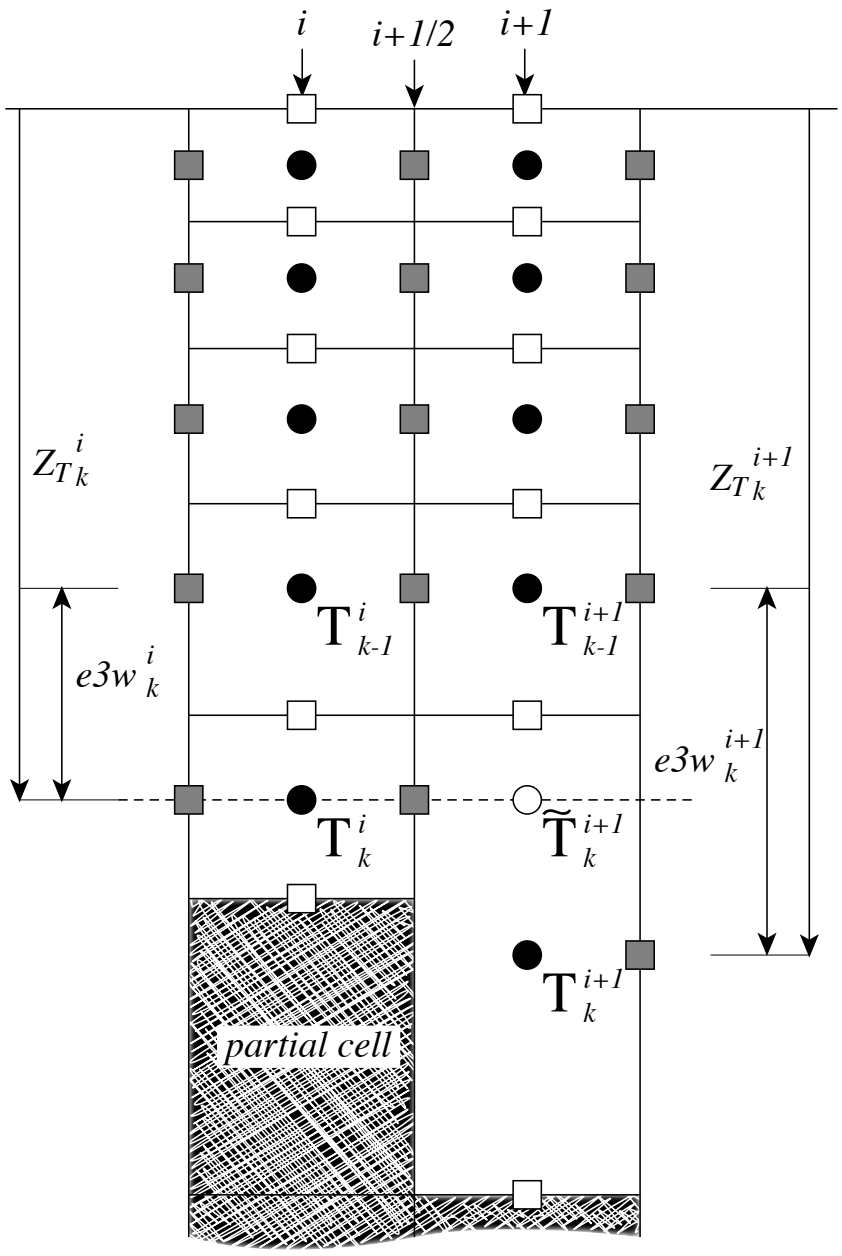

FIG. 5.5 – Discretisation of the horizontal difference and average of tracers in the z-partial step coordinate (*ln_zps*=true) in the case $(e3w_k^{i+1} - e3w_k^i) > 0$. A linear interpolation is used to estimate $\widetilde{T}_k^{i+1}$, the tracer value at the depth of the shallower tracer point of the two adjacent bottom T-points. The horizontal difference is then given by : $\delta_{i+1/2}T_k = \widetilde{T}_k^{i+1} - T_k^i$ and the average by : $\overline{T}_k^{i+1/2} = (\widetilde{T}_k^{i+1/2} - T_k^i)/2$.

is applied for advection. The main motivation is to preserve the domain averaged mean variance of the advected field when using the 2^{nd} order centred scheme. Sensitivity of the advection schemes to the way horizontal averages are performed in the vicinity of partial cells should be further investigated in the near future.

6 Ocean Dynamics (DYN)

Contents

Using the representation described in Chapter 4, several semi-discrete space forms of the dynamical equations are available depending on the vertical coordinate used and on the conservation properties of the vorticity term. In all the equations presented here, the masking has been omitted for simplicity. One must be aware that all the quantities are masked fields and that each time an average or difference operator is used, the resulting field is multiplied by a mask.

The prognostic ocean dynamics equation can be summarized as follows :

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

NXT stands for next, referring to the time-stepping. The first group of terms on the rhs of this equation corresponds to the Coriolis and advection terms that are decomposed into either a vorticity part (VOR), a kinetic energy part (KEG) and a vertical advection part (ZAD) in the vector invariant formulation, or a Coriolis and advection part (COR+ADV) in the flux formulation. The terms following these are the pressure gradient contributions (HPG, Hydrostatic Pressure Gradient, and SPG, Surface Pressure Gradient) ; and contributions from lateral diffusion (LDF) and vertical diffusion (ZDF), which are added to the rhs in the *dynldf.F90* and *dynzdf.F90* modules. The vertical diffusion term includes the surface and bottom stresses. The external forcings and parameterisations require complex inputs (surface wind stress calculation using bulk formulae, estimation of mixing coefficients) that are carried out in modules SBC, LDF and ZDF and are described in Chapters 7, 9 and 10, respectively.

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

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

The user has the option of extracting and outputting each tendency term from the 3D momentum equations (**key_trddyn** defined), as described in Chap.13. Furthermore, the tendency terms associated with the 2D barotropic vorticity balance (when **key_trdvor** is defined) can be derived from the 3D terms.

6.1 Sea surface height and diagnostic variables (η, ζ, χ, w)

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

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

$$\zeta = \frac{1}{e_{1f}\, e_{2f}} \left(\delta_{i+1/2} \left[e_{2v}\, v \right] - \delta_{j+1/2} \left[e_{1u}\, u \right] \right) \tag{6.1}$$

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

$$\chi = \frac{1}{e_{1t}\, e_{2t}\, e_{3t}} \left(\delta_i \left[e_{2u}\, e_{3u}\, u \right] + \delta_j \left[e_{1v}\, e_{3v}\, v \right] \right) \tag{6.2}$$

Note that although the vorticity has the same discrete expression in z- and s-coordinates, 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 not necessarily defined at the same depth).

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

6.1.2 Horizontal divergence and relative vorticity (*sshwzv.F90*)

The sea surface height is given by :

$$\begin{aligned} \frac{\partial \eta}{\partial t} &\equiv \frac{1}{e_{1t} e_{2t}} \sum_k \left\{ \delta_i \left[e_{2u}\, e_{3u}\, u \right] + \delta_j \left[e_{1v}\, e_{3v}\, v \right] \right\} - \frac{emp}{\rho_w} \\ &\equiv \sum_k \chi\, e_{3t} - \frac{emp}{\rho_w} \end{aligned} \tag{6.3}$$

where *emp* is the surface freshwater budget (evaporation minus precipitation), expressed in Kg/m^2/s (which is equal to mm/s), and ρ_w=1,035 Kg/m^3 is the reference density of sea water (Boussinesq approximation). If river runoff is expressed as a surface freshwater flux (see §7) then *emp* can be written as the evaporation minus precipitation, minus the river runoff. The sea-surface height is evaluated using exactly the same time stepping scheme as the tracer equation (5.25) : a leapfrog scheme in combination with an Asselin time filter, $i.e.$ the velocity appearing in (6.3) is centred in time (*now* velocity). This is of paramount importance. Replacing T by the number 1 in the tracer equation and summing over the water column must lead to the sea surface height equation otherwise tracer content will not be conserved **??**.

The vertical velocity is computed by an upward integration of the horizontal divergence starting at the bottom, taking into account the change of the thickness of the levels :

$$\begin{cases} w|_{k_b-1/2} = 0 & \text{where } k_b \text{ is the level just above the sea floor} \\ w|_{k+1/2} = w|_{k-1/2} + e_{3t}|_k \ \chi|_k - \dfrac{1}{2\Delta t}\left(e_{3t}^{t+1}|_k - e_{3t}^{t-1}|_k\right) \end{cases} \tag{6.4}$$

In the case of a non-linear free surface (**key_vvl**), the top vertical velocity is $-emp/\rho_w$, as changes in the divergence of the barotropic transport are absorbed into the change of the level thicknesses, re-orientated downward. In the case of a linear free surface, the time derivative in (6.4) disappears. The upper boundary condition applies at a fixed level $z = 0$. The top vertical velocity is thus equal to the divergence of the barotropic transport ($i.e.$ the first term in the right-hand-side of (6.3)).

Note also that whereas the vertical velocity has the same discrete expression in z- and s-coordinates, its physical meaning is not the same : in the second case, w is the velocity normal to the s-surfaces. Note also that the k-axis is re-orientated downwards in the FORTRAN code compared to the indexing used in the semi-discrete equations such as (6.4) (see §4.1.3).

6.2 Coriolis and Advection : vector invariant form

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

The vector invariant form of the momentum equations is the one most often used in applications of the *NEMO* ocean model. The flux form option (see next section) has been present since version 2. Coriolis and momentum advection terms are evaluated using a leapfrog scheme, $i.e.$ the velocity appearing in these expressions is centred in time (*now* velocity). At the lateral boundaries either free slip, no slip or partial slip boundary conditions are applied following Chap.8.

6.2.1 Vorticity term (*dynvor.F90*)

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

Four discretisations of the vorticity term (*ln_dynvor_xxx*=true) are available : conserving potential enstrophy of horizontally non-divergent flow (ENS scheme) ; conserving horizontal kinetic energy (ENE scheme) ; conserving potential enstrophy for the relative vorticity term and horizontal kinetic energy for the planetary vorticity term (MIX scheme) ; or conserving both the potential enstrophy of horizontally non-divergent flow and horizontal kinetic energy (ENE scheme) (see Appendix **??**). The vorticity terms are all computed in dedicated routines that can be found in the *dynvor.F90* module.

Enstrophy conserving scheme (*ln_dynvor_ens*=true)

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

$$\begin{cases} +\dfrac{1}{e_{1u}} \overline{\left(\dfrac{\zeta+f}{e_{3f}}\right)}^{i} & \overline{\overline{(e_{1v}\, e_{3v}\, v)}}^{i,j+1/2} \\ -\dfrac{1}{e_{2v}} \overline{\left(\dfrac{\zeta+f}{e_{3f}}\right)}^{j} & \overline{\overline{(e_{2u}\, e_{3u}\, u)}}^{i+1/2,j} \end{cases} \tag{6.5}$$

Energy conserving scheme (*ln_dynvor_ene*=true)

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

$$\begin{cases} +\dfrac{1}{e_{1u}} \overline{\left(\dfrac{\zeta+f}{e_{3f}}\right) \overline{(e_{1v}\, e_{3v}\, v)}^{i+1/2}}^{j} \\ -\dfrac{1}{e_{2v}} \overline{\left(\dfrac{\zeta+f}{e_{3f}}\right) \overline{(e_{2u}\, e_{3u}\, u)}^{j+1/2}}^{i} \end{cases} \tag{6.6}$$

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

For the mixed energy/enstrophy conserving scheme (MIX scheme), a mixture of the two previous schemes is used. It consists of the ENS scheme (C.13) for the relative vorticity term, and of the ENE scheme (6.6) applied to the planetary vorticity term.

$$\begin{cases} +\dfrac{1}{e_{1u}} \overline{\left(\dfrac{\zeta}{e_{3f}}\right)}^{i} \overline{\overline{(e_{1v}\, e_{3v}\, v)}}^{i,j+1/2} - \dfrac{1}{e_{1u}} \overline{\left(\dfrac{f}{e_{3f}}\right) \overline{(e_{1v}\, e_{3v}\, v)}^{i+1/2}}^{j} \\ -\dfrac{1}{e_{2v}} \overline{\left(\dfrac{\zeta}{e_{3f}}\right)}^{j} \overline{\overline{(e_{2u}\, e_{3u}\, u)}}^{i+1/2,j} + \dfrac{1}{e_{2v}} \overline{\left(\dfrac{f}{e_{3f}}\right) \overline{(e_{2u}\, e_{3u}\, u)}^{j+1/2}}^{i} \end{cases} \tag{6.7}$$

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

In both the ENS and ENE schemes, it is apparent that the combination of i and j averages of the velocity allows for the presence of grid point oscillation structures that will be invisible to the operator. These structures are *computational modes* that will be at least partly damped by the momentum diffusion operator ($i.e.$ the subgrid-scale advection), but not by the resolved advection term. The ENS and ENE schemes therefore do not contribute to dump any grid point noise in the horizontal velocity field. Such noise would result in more noise in the vertical velocity field, an undesirable feature. This is a well-known characteristic of C-grid discretization where u and v are located at different grid

points, a price worth paying to avoid a double averaging in the pressure gradient term as in the B-grid.

A very nice solution to the problem of double averaging was proposed by **?**. The idea is to get rid of the double averaging by considering triad combinations of vorticity. It is noteworthy that this solution is conceptually quite similar to the one proposed by [**?**] for the discretization of the iso-neutral diffusion operator (see App.E).

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

$$q = \frac{\zeta + f}{e_{3f}} \tag{6.8}$$

where the relative vorticity is defined by (6.1), the Coriolis parameter is given by $f = 2\,\Omega\,\sin\varphi_f$ and the layer thickness at f-points is :

$$e_{3f} = \overline{\overline{e_{3t}}}^{\,i+1/2,j+1/2} \tag{6.9}$$

Note that a key point in (6.9) is that the averaging in the **i**- and **j**- directions uses the masked vertical scale factor but is always divided by 4, not by the sum of the masks at the four T-points. 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} into the land areas. This feature is essential for the z-coordinate with partial steps.

Next, the vorticity triads, ${}^{i}_{j}\mathbb{Q}^{i_p}_{j_p}$ can be defined at a T-point as the following triad combinations of the neighbouring potential vorticities defined at f-points (Fig. 6.1) :

$${}^{j}_{i}\mathbb{Q}^{i_p}_{j_p} = \frac{1}{12}\left(q^{i-i_p}_{j+j_p} + q^{i+j_p}_{j+i_p} + q^{i+i_p}_{j-j_p}\right) \tag{6.10}$$

where the indices i_p and k_p take the values : $i_p = -1/2$ or $1/2$ and $j_p = -1/2$ or $1/2$.

Finally, the vorticity terms are represented as :

$$\begin{cases} +q\,e_3\,v \equiv +\dfrac{1}{e_{1u}} \displaystyle\sum_{i_p,\,k_p} {}^{i+1/2-i_p}_{\quad j}\mathbb{Q}^{i_p}_{j_p}\,(e_{1v}\,e_{3v}\,v)^{i+1/2-i_p}_{j+j_p} \\ -q\,e_3\,u \equiv -\dfrac{1}{e_{2v}} \displaystyle\sum_{i_p,\,k_p} {}^{\quad i}_{j+1/2-j_p}\mathbb{Q}^{i_p}_{j_p}\,(e_{2u}\,e_{3u}\,u)^{i+i_p}_{j+1/2-j_p} \end{cases} \tag{6.11}$$

This EEN scheme in fact combines the conservation properties of the ENS and ENE schemes. It conserves both total energy and potential enstrophy in the limit of horizontally nondivergent flow (*i.e.* χ=0) (see Appendix **??**). Applied to a realistic ocean configuration, it has been shown that it leads to a significant reduction of the noise in the vertical velocity field [**?**]. Furthermore, used in combination with a partial steps representation of bottom topography, it improves the interaction between current and topography, leading to a larger topostrophy of the flow [**??**].

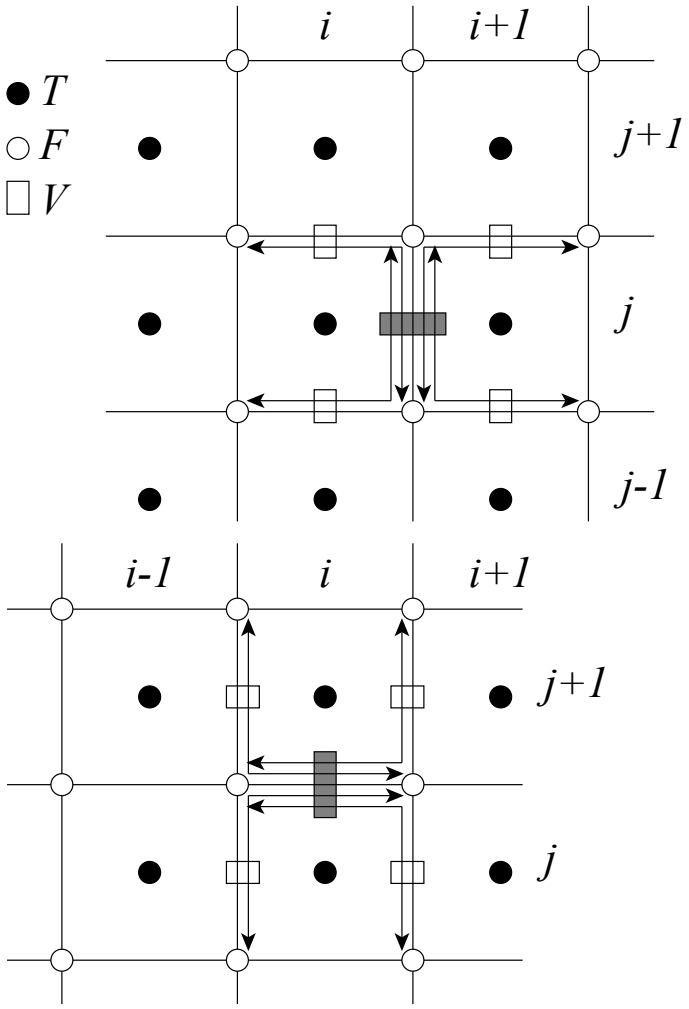

FIG. 6.1 – Triads used in the energy and enstrophy conserving scheme (een) for u-component (upper panel) and v-component (lower panel).

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

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

$$\begin{cases} -\dfrac{1}{2\,e_{1u}}\,\delta_{i+1/2}\left[\overline{u^2}^{\,i}+\overline{v^2}^{\,j}\right] \\ -\dfrac{1}{2\,e_{2v}}\,\delta_{j+1/2}\left[\overline{u^2}^{\,i}+\overline{v^2}^{\,j}\right] \end{cases} \tag{6.12}$$

6.2.3 Vertical advection term (*dynzad.F90*)

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

$$\begin{cases} -\dfrac{1}{e_{1u}\, e_{2u}\, e_{3u}} \; \overline{\overline{e_{1t}\, e_{2t}\, w}^{\,i+1/2}\; \delta_{k+1/2}\left[u\right]}^{\,k} \\ -\dfrac{1}{e_{1v}\, e_{2v}\, e_{3v}} \; \overline{\overline{e_{1t}\, e_{2t}\, w}^{\,j+1/2}\; \delta_{k+1/2}\left[u\right]}^{\,k} \end{cases} \tag{6.13}$$

6.3 Coriolis and Advection : flux form

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

In the flux form (as in the vector invariant form), the Coriolis and momentum advection terms are evaluated using a leapfrog scheme, $i.e.$ the velocity appearing in their expressions is centred in time (*now* velocity). At the lateral boundaries either free slip, no slip or partial slip boundary conditions are applied following Chap.8.

6.3.1 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 :

$$\begin{aligned} f + \frac{1}{e_1 e_2}\left(v\frac{\partial e_2}{\partial i} - u\frac{\partial e_1}{\partial j} \right) \\ \equiv f + \frac{1}{e_{1f} e_{2f}}\left(\overline{v}^{\,i+1/2}\delta_{i+1/2}\left[e_{2u}\right] - \overline{u}^{\,j+1/2}\delta_{j+1/2}\left[e_{1u}\right] \right) \end{aligned} \tag{6.14}$$

Any of the (C.13), (6.6) and (C.15) schemes can be used to compute the product of the Coriolis parameter and the vorticity. However, the energy-conserving scheme (C.15) has exclusively been used to date. This term is evaluated using a leapfrog scheme, $i.e.$ the velocity is centred in time (*now* velocity).

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

The discrete expression of the advection term is given by :

$$\begin{cases} \dfrac{1}{e_{1u}\,e_{2u}\,e_{3u}} \Big(\delta_{i+1/2}\left[\overline{e_{2u}\,e_{3u}\,u}^{\,i}\;u_t \right] + \delta_j \left[\overline{e_{1u}\,e_{3u}\,v}^{\,i+1/2}\;u_f \right] \\ \qquad\qquad + \delta_k \left[\overline{e_{1w}\,e_{2w}\,w}^{\,i+1/2}\;u_{uw} \right] \Big) \\ \dfrac{1}{e_{1v}\,e_{2v}\,e_{3v}} \Big(\delta_i \left[\overline{e_{2u}\,e_{3u}\,u}^{\,j+1/2}\;v_f \right] + \delta_{j+1/2}\left[\overline{e_{1u}\,e_{3u}\,v}^{\,i}\;v_t \right] \\ \qquad\qquad + \delta_k \left[\overline{e_{1w}\,e_{2w}\,w}^{\,j+1/2}\;v_{vw} \right] \Big) \end{cases} \tag{6.15}$$

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

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

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

$$\begin{cases} u_T^{cen2} = \overline{u}^{\,i} & u_F^{cen2} = \overline{u}^{\,j+1/2} & u_{uw}^{cen2} = \overline{u}^{\,k+1/2} \\ v_F^{cen2} = \overline{v}^{\,i+1/2} & v_F^{cen2} = \overline{v}^{\,j} & v_{vw}^{cen2} = \overline{v}^{\,k+1/2} \end{cases} \tag{6.16}$$

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

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

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

$$u_T^{ubs} = \overline{u}^{\,i} - \frac{1}{6} \begin{cases} u"_{i-1/2} & \text{if } \overline{e_{2u}\,e_{3u}\,u}^{\,i} \geqslant 0 \\ u"_{i+1/2} & \text{if } \overline{e_{2u}\,e_{3u}\,u}^{\,i} < 0 \end{cases} \tag{6.17}$$

where $u"_{i+1/2} = \delta_{i+1/2}\left[\delta_i\left[u\right]\right]$. This results in a dissipatively dominant ($i.e.$ hyper-diffusive) truncation error [**?**]. The overall performance of the advection scheme is similar

to that reported in **?**. It is a relatively good compromise between accuracy and smoothness. It is not a *positive* scheme, meaning that false extrema are permitted. But the amplitudes of the false extrema are significantly reduced over those in the centred second order method. As the scheme already includes a diffusion component, it can be used without explicit lateral diffusion on momentum ($i.e.$ *ln_dynldf_lap*=*ln_dynldf_bilap*=false), and it is recommended to do so.

The UBS scheme is not used in all directions. In the vertical, the centred 2^{nd} order evaluation of the advection is preferred, $i.e.$ u^{ubs}_{uw} and u^{ubs}_{vw} in (6.16) are used. UBS is diffusive and is associated with vertical mixing of momentum.

For stability reasons, the first term in (6.17), which corresponds to a second order centred scheme, is evaluated using the *now* velocity (centred in time), while the second term, which is the diffusion part of the scheme, is evaluated using the *before* velocity (forward in time). This is discussed by **?** in the context of the Quick advection scheme.

Note that the UBS and QUICK (Quadratic Upstream Interpolation for Convective Kinematics) schemes only differ by one coefficient. Replacing $1/6$ by $1/8$ in (6.17) leads to the QUICK advection scheme [**?**]. This option is not available through a namelist parameter, since the $1/6$ coefficient is hard coded. Nevertheless it is quite easy to make the substitution in the *dynadv_ubs.F90* module and obtain a QUICK scheme.

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

6.4 Hydrostatic pressure gradient (*dynhpg.F90*)

```
!-----------------------------------------------------------------------
&namdyn_hpg    !   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)
   rn_gamma    = 0.e0      !  weighting coefficient (wdj scheme)
   ln_dynhpg_imp = .false. !  time stepping: semi-implicit time scheme  (T)
                               !           centered      time scheme  (F)
/
```

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

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

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

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

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

$$\begin{cases} \delta_{i+1/2}\left[p^h\right]\Big|_{k=km} = \dfrac{1}{2} g \;\; \delta_{i+1/2}\left[e_{3w}\;\rho\right]\Big|_{k=km} \\ \delta_{j+1/2}\left[p^h\right]\Big|_{k=km} = \dfrac{1}{2} g \;\; \delta_{j+1/2}\left[e_{3w}\;\rho\right]\Big|_{k=km} \end{cases} \tag{6.18}$$

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

$$\begin{cases} \delta_{i+1/2}\left[p^h\right]\Big|_{k} = \delta_{i+1/2}\left[p^h\right]\Big|_{k-1} + \dfrac{1}{2}\; g \;\; \delta_{i+1/2}\left[e_{3w}\;\overline{\rho}^{k+1/2}\right]\Big|_{k} \\ \delta_{j+1/2}\left[p^h\right]\Big|_{k} = \delta_{j+1/2}\left[p^h\right]\Big|_{k-1} + \dfrac{1}{2}\; g \;\; \delta_{j+1/2}\left[e_{3w}\;\overline{\rho}^{k+1/2}\right]\Big|_{k} \end{cases} \tag{6.19}$$

Note that the $1/2$ factor in (6.18) is adequate because of the definition of e_{3w} as the vertical derivative of the scale factor at the surface level ($z = 0$). Note also that in case of variable volume level (**key_vvl** defined), the surface pressure gradient is included in (6.18) and (6.19) through the space and time variations of the vertical scale factor e_{3w}.

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

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

Apart from this modification, the horizontal hydrostatic pressure gradient evaluated in the z-coordinate with partial step is exactly as in the pure z-coordinate case. As explained in detail in section §5.9, the nonlinearity of pressure effects in the equation of state is such that it is better to interpolate temperature and salinity vertically before computing the density. Horizontal gradients of temperature and salinity are needed for the TRA modules, which is the reason why the horizontal gradients of density at the deepest model level are computed in module *zpsdhe.F90* located in the TRA directory and described in §5.9.

6.4.3 s- and z-s-coordinates

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

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

$$\begin{cases} -\dfrac{1}{\rho_o\, e_{1u}}\, \delta_{i+1/2}\left[p^h\right] + \dfrac{g\, \overline{\rho}^{i+1/2}}{\rho_o\, e_{1u}}\, \delta_{i+1/2}\left[z_t\right] \\ -\dfrac{1}{\rho_o\, e_{2v}}\, \delta_{j+1/2}\left[p^h\right] + \dfrac{g\, \overline{\rho}^{j+1/2}}{\rho_o\, e_{2v}}\, \delta_{j+1/2}\left[z_t\right] \end{cases} \tag{6.20}$$

Where the first term is the pressure gradient along coordinates, computed as in (6.18) - (6.19), and z_T is the depth of the T-point evaluated from the sum of the vertical scale factors at the w-point (e_{3w}). The version *ln_dynhpg_hel*=true has been added by Aike Beckmann and involves a redefinition of the relative position of T-points relative to w-points.

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

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

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

Note that expression (6.20) is used when the variable volume formulation is activated (**key_vvl**) because in that case, even with a flat bottom, the coordinate surfaces are not horizontal but follow the free surface [**?**]. The other pressure gradient options are not yet available.

6.4.4 Time-scheme (*ln_dynhpg_imp*= true/false)

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

• leapfrog scheme (*ln_dynhpg_imp*=true) :

$$\frac{u^{t+\Delta t} - u^{t-\Delta t}}{2\Delta t} = \cdots - \frac{1}{\rho_o\, e_{1u}} \delta_{i+1/2}\left[p_h^t\right] \tag{6.21}$$

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

$$\frac{u^{t+\Delta t} - u^{t-\Delta t}}{2\Delta t} = \cdots - \frac{1}{4\,\rho_o\, e_{1u}} \delta_{i+1/2}\left[p_h^{t+\Delta t} + 2\,p_h^t + p_h^{t-\Delta t}\right] \tag{6.22}$$

The semi-implicit time scheme (6.22) is made possible without significant additional computation since the density can be updated to time level $t+\Delta t$ before computing the horizontal hydrostatic pressure gradient. It can be easily shown that the stability limit associated with the hydrostatic pressure gradient doubles using (6.22) compared to that

using the standard leapfrog scheme (6.21). Note that (6.22) is equivalent to applying a time filter to the pressure gradient to eliminate high frequency IGWs. Obviously, when using (6.22), the doubling of the time-step is achievable only if no other factors control the time-step, such as the stability limits associated with advection or diffusion.

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

$$\rho^t = \rho(\widetilde{T}, \widetilde{S}, z_t) \quad \text{with} \quad \widetilde{X} = 1/4 \left(X^{t+\Delta t} + 2\, X^t + X^{t-\Delta t} \right) \tag{6.23}$$

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

6.5 Surface pressure gradient (*dynspg.F90*)

```
!------------------------------------------------------------------------
!namdyn_spg    !   surface pressure gradient    (CPP key only)
!------------------------------------------------------------------------
!                          !  explicit free surface                    ("key_dynspg_exp")
!                          !  filtered free surface                    ("key_dynspg_flt")
!                          !  split-explicit free surface              ("key_dynspg_ts")
```

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) and the variable volume case (nonlinear free surface, **key_vvl** is defined). In the linear free surface case (§2.2.2) the vertical scale factors e_3 are fixed in time, while they are time-dependent in the nonlinear case (§2.2.2). 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*.

The form of the surface pressure gradient term depends on how the user wants to handle the fast external gravity waves that are a solution of the analytical equation (§2.2). Three formulations are available, all controlled by a CPP key (ln_dynspg_xxx) : an explicit formulation which requires a small time step ; a filtered free surface formulation which allows a larger time step by adding a filtering term into the momentum equation ; and a split-explicit free surface formulation, described below, which also allows a larger time step.

The extra term introduced in the filtered method is calculated implicitly, so that a solver is used to compute it. As a consequence the update of the $next$ velocities is done in module *dynspg_flt.F90* and not in *dynnxt.F90*.

6.5.1 Explicit free surface (key_dynspg_exp)

In the explicit free surface formulation (**key_dynspg_exp** defined), the model time step is chosen to be small enough to resolve the external gravity waves (typically a few tens of seconds). The surface pressure gradient, evaluated using a leap-frog scheme ($i.e.$ centered in time), is thus simply given by :

$$\begin{cases} -\dfrac{1}{e_{1u}\,\rho_o}\,\delta_{i+1/2}\left[\rho\,\eta\right] \\ -\dfrac{1}{e_{2v}\,\rho_o}\,\delta_{j+1/2}\left[\rho\,\eta\right] \end{cases} \tag{6.24}$$

Note that in the non-linear free surface case ($i.e.$ **key_vvl** defined), the surface pressure gradient is already included in the momentum tendency through the level thickness variation allowed in the computation of the hydrostatic pressure gradient. Thus, nothing is done in the *dynspg_exp.F90* module.

6.5.2 Split-Explicit free surface (key_dynspg_ts)

The split-explicit free surface formulation used in *NEMO* (**key_dynspg_ts** defined), also called the time-splitting formulation, follows the one proposed by **?**. The general idea is to solve the free surface equation and the associated barotropic velocity equations with a smaller time step than Δt, the time step used for the three dimensional prognostic variables (Fig. 6.2). The size of the small time step, Δt_e (the external mode or barotropic time step) is provided through the *nn_baro* namelist parameter as : $\Delta t_e = \Delta t / nn\_baro$.

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

6.5.3 Filtered free surface (key_dynspg_flt)

The filtered formulation follows the **?** implementation. The extra term introduced in the equations (see §2.2.2) is solved implicitly. The elliptic solvers available in the code are documented in §13.

Note that in the linear free surface formulation (**key_vvl** not defined), the ocean depth is time-independent and so is the matrix to be inverted. It is computed once and for all and applies to all ocean time steps.

6.6 Lateral diffusion term (*dynldf.F90*)

```
!-----------------------------------------------------------------------
&namdyn_ldf    !   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
```

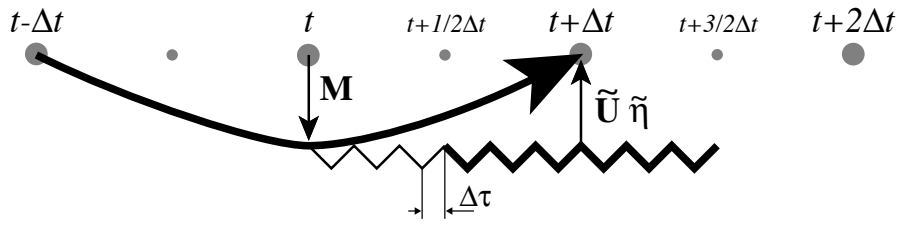

FIG. 6.2 – Schematic of the split-explicit time stepping scheme for the external and internal modes. Time increases to the right. Internal mode time steps (which are also the model time steps) are denoted by $t-\Delta t$, t, $t+\Delta t$, and $t+2\Delta t$. The curved line represents a leap-frog time step, and the smaller time steps $N\Delta t_e = \frac{3}{2}\Delta t$ are denoted by the zig-zag line. The vertically integrated forcing **M**(t) computed at the model time step t represents the interaction between the external and internal motions. While keeping **M** and freshwater forcing field fixed, a leap-frog integration carries the external mode variables (surface height and vertically integrated velocity) from t to $t + \frac{3}{2}\Delta t$ using N external time steps of length Δt_e. Time averaging the external fields over the $\frac{2}{3}N + 1$ time steps (endpoints included) centers the vertically integrated velocity and the sea surface height at the model timestep $t + \Delta t$. These averaged values are used to update **M**(t) with both the surface pressure gradient and the Coriolis force, therefore providing the $t + \Delta t$ velocity. The model time stepping scheme can then be achieved by a baroclinic leap-frog time step that carries the surface height from $t - \Delta t$ to $t + \Delta t$.

```
   ln_dynldf_hor    =  .true.   !  horizontal (geopotential)           (require "key_ldfslp" in s-coord.)
   ln_dynldf_iso    =  .false.  !  iso-neutral                        (require "key_ldfslp")
   !                        !  Coefficient
   rn_ahm_0_lap     = 40000.    !  horizontal laplacian eddy viscosity   [m2/s]
   rn_ahmb_0        =     0.    !  background eddy viscosity for ldf_iso [m2/s]
   rn_ahm_0_blp     =     0.    !  horizontal bilaplacian eddy viscosity [m4/s]
/
```

The options available for lateral diffusion are to use either 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.9). 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 §**??**)

At the lateral boundaries either free slip, no slip or partial slip boundary conditions are applied according to the user's choice (see Chap.8).

6.6.1 Iso-level laplacian operator (*ln_dynldf_lap*=true)

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

$$\begin{cases} D_u^{l\mathbf{U}} = \dfrac{1}{e_{1u}} \delta_{i+1/2} \left[A_T^{lm} \, \chi \right] - \dfrac{1}{e_{2u}\, e_{3u}} \delta_j \left[A_f^{lm} \, e_{3f} \zeta \right] \\ \\ D_v^{l\mathbf{U}} = \dfrac{1}{e_{2v}} \delta_{j+1/2} \left[A_T^{lm} \, \chi \right] + \dfrac{1}{e_{1v}\, e_{3v}} \delta_i \left[A_f^{lm} \, e_{3f} \zeta \right] \end{cases} \tag{6.25}$$

As explained in §2.5.2, 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 of the momentum diffusion.

6.6.2 Rotated laplacian operator (*ln_dynldf_iso*=true)

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

symmetry. The resulting discrete representation is :

$$
\begin{aligned}
D_u^{l\mathbf{U}} = & \frac{1}{e_{1u}\, e_{2u}\, e_{3u}} \\
& \left\{ \quad \delta_{i+1/2}\left[A_T^{lm}\left(\frac{e_{2t}\, e_{3t}}{e_{1t}}\, \delta_i[u] - e_{2t}\, r_{1t}\, \overline{\overline{\delta_{k+1/2}[u]}}^{\,i,\,k} \right) \right] \right. \\
& \quad + \delta_j\left[A_f^{lm}\left(\frac{e_{1f}\, e_{3f}}{e_{2f}}\, \delta_{j+1/2}[u] - e_{1f}\, r_{2f}\, \overline{\overline{\delta_{k+1/2}[u]}}^{\,j+1/2,\,k} \right) \right] \\
& \quad + \delta_k\left[A_{uw}^{lm}\left(-e_{2u}\, r_{1uw}\, \overline{\overline{\delta_{i+1/2}[u]}}^{\,i+1/2,\,k+1/2} \right.\right. \\
& \qquad\qquad - e_{1u}\, r_{2uw}\, \overline{\overline{\delta_{j+1/2}[u]}}^{\,j,\,k+1/2} \\
& \qquad\qquad \left.\left. + \frac{e_{1u}\, e_{2u}}{e_{3uw}}\, \left(r_{1uw}^2 + r_{2uw}^2\right)\, \delta_{k+1/2}[u] \right) \right] \quad \left. \right\}
\end{aligned}
\tag{6.26}
$$

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

where r_1 and r_2 are the slopes between the surface along which the diffusion operator acts and the surface of computation (z- or s-surfaces). The way these slopes are evaluated is given in the lateral physics chapter (Chap.9).

6.6.3 Iso-level bilaplacian operator (*ln_dynldf_bilap*=true)

The lateral fourth order operator formulation on momentum is obtained by applying (6.25) 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.8).

6.7 Vertical diffusion term (*dynzdf.F90*)

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

```
!-----------------------------------------------------------------------
   rn_avm0     =   1.2e-4  !  vertical eddy viscosity   [m2/s]          (background Kz if not "key_zdfcst")
   rn_avt0     =   1.2e-5  !  vertical eddy diffusivity [m2/s]          (background Kz if not "key_zdfcst")
   nn_avb      =    0      !  profile for background avt & avm (=1) or not (=0)
   nn_havtb    =    0      !  horizontal shape for avtb (=1) or not (=0)
   ln_zdfevd   = .true.    !  enhanced vertical diffusion (evd) (T) or not (F)
   nn_evdm     =    0      !  evd apply on tracer (=0) or on tracer and momentum (=1)
   rn_avevd    =  100.     !  evd mixing coefficient [m2/s]
   ln_zdfnpc   = .false.   !  Non-Penetrative Convective algorithm (T) or not (F)
   nn_npc      =    1           !  frequency of application of npc
   nn_npcp     =  365           !  npc control print frequency
   ln_zdfexp   = .false.   !  time-stepping: split-explicit (T) or implicit (F) time stepping
   nn_zdfexp   =    3           !  number of sub-timestep for ln_zdfexp=T
/
```

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

The formulation of the vertical subgrid scale physics is the same whatever the vertical coordinate is. The vertical diffusion operators given by (2.36) take the following semi-discrete space form :

$$\begin{cases} D_u^{vm} \equiv \dfrac{1}{e_{3u}}\,\delta_k\left[\dfrac{A_{uw}^{vm}}{e_{3uw}}\,\delta_{k+1/2}[\,u\,]\right] \\[2ex] D_v^{vm} \equiv \dfrac{1}{e_{3v}}\,\delta_k\left[\dfrac{A_{vw}^{vm}}{e_{3vw}}\,\delta_{k+1/2}[\,v\,]\right] \end{cases} \tag{6.27}$$

where A_{uw}^{vm} and A_{vw}^{vm} are the vertical eddy viscosity and diffusivity coefficients. The way these coefficients are evaluated depends on the vertical physics used (see §10).

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

$$\left(\frac{A^{vm}}{e_3}\,\frac{\partial \mathbf{U}_h}{\partial k}\right)\Bigg|_{z=1} = \frac{1}{\rho_o}\begin{pmatrix}\tau_u\\ \tau_v\end{pmatrix} \tag{6.28}$$

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.7)

The turbulent flux of momentum at the bottom of the ocean is specified through a bottom friction parameterisation (see §10.4)

6.8 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. Another forcing term is the tidal potential. Both of which will be introduced into the reference version soon.

6.9 Time evolution term (*dynnxt.F90*)

```
!-----------------------------------------------------------------------
&namdom        !   space and time domain (bathymetry, mesh, timestep)
!-----------------------------------------------------------------------
   nn_bathy    =    1      !  compute (=0) or read (=1) the bathymetry file
   nn_closea    =   0      !  remove (=0) or keep (=1) closed seas and lakes (ORCA)
   nn_msh      =    0      !  create (=1) a mesh file or not (=0)
   rn_hmin     =   -3.     !  min depth of the ocean (>0) or min number of ocean level (<0)
   rn_e3zps_min=   20.     !  partial step thickness is set larger than the minimum of
   rn_e3zps_rat=    0.1    !  rn_e3zps_min and rn_e3zps_rat*e3t, with 0<rn_e3zps_rat<1
                           !
   rn_rdt      = 5760.     !  time step for the dynamics (and tracer if nn_acc=0)
   nn_baro     =   64      !  number of barotropic time step            ("key_dynspg_ts")
   rn_atfp     =    0.1    !  asselin time filter parameter
   nn_acc      =    0      !  acceleration of convergence : =1      used, rdt < rdttra(k)
                              !                             =0, not used, rdt = rdttra
   rn_rdtmin   = 28800.        !  minimum time step on tracers (used if nn_acc=1)
   rn_rdtmax   = 28800.        !  maximum time step on tracers (used if nn_acc=1)
   rn_rdth     =  800.         !  depth variation of tracer time step  (used if nn_acc=1)
/
```

The general framework for dynamics time stepping is a leap-frog scheme, $i.e.$ a three level centred time scheme associated with an Asselin time filter (cf. Chap.3). The scheme is applied to the velocity, except when using the flux form of momentum advection (cf. §6.3) in the variable volume case (**key_vvl** defined), where it has to be applied to the thickness weighted velocity (see §A.3)

• vector invariant form or linear free surface (*ln_dynhpg_vec*=true ; **key_vvl** not defined) :

$$\begin{cases} u^{t+\Delta t} = u_f^{t-\Delta t} + 2\Delta t \, \text{RHS}_u^t \\ u_f^t \quad = u^t + \gamma \left[u_f^{t-\Delta t} - 2u^t + u^{t+\Delta t} \right] \end{cases} \tag{6.29}$$

• flux form and nonlinear free surface (*ln_dynhpg_vec*=false ; **key_vvl** defined) :

$$\begin{cases} (e_{3u}\, u)^{t+\Delta t} = (e_{3u}\, u)_f^{t-\Delta t} + 2\Delta t \, e_{3u} \, \text{RHS}_u^t \\ (e_{3u}\, u)_f^t \quad = (e_{3u}\, u)^t + \gamma \left[(e_{3u}\, u)_f^{t-\Delta t} - 2\,(e_{3u}\, u)^t + (e_{3u}\, u)^{t+\Delta t} \right] \end{cases} \tag{6.30}$$

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 *nn_atfp* (namelist parameter). Its default value is *nn_atfp* = 10^{-3}. In both cases, the modified Asselin filter is not applied since perfect conservation is not an issue for the momentum equations.

Note that with the filtered free surface, the update of the *after* velocities is done in the *dynsp_flt.F90* module, and only array swapping and Asselin filtering is done in *dynnxt.F90*.

7 Surface Boundary Condition (SBC)

Contents

```
!-----------------------------------------------------------------------
&namsbc        !   Surface Boundary Condition (surface module)
!-----------------------------------------------------------------------
   nn_fsbc     = 5         !  frequency of surface boundary condition computation
                           !     (also = the frequency of sea-ice model call)
   ln_ana      = .false.   !  analytical formulation                    (T => fill namsbc_ana )
   ln_flx      = .false.   !  flux formulation                          (T => fill namsbc_flx )
   ln_blk_clio = .false.   !  CLIO bulk formulation                     (T => fill namsbc_clio)
   ln_blk_core = .true.    !  CORE bulk formulation                     (T => fill namsbc_core)
   ln_cpl      = .false.   !  Coupled formulation                       (T => fill namsbc_cpl )
   ln_apr_dyn  = .false.   !  Patm gradient added in ocean & ice Eqs.   (T => fill namsbc_apr )
   nn_ice      = 2         !  =0 no ice boundary condition   ,
                           !  =1 use observed ice-cover      ,
                           !  =2 ice-model used                        ("key_lim3" or "key_lim2)
   ln_dm2dc    = .false.   !  daily mean to diurnal cycle on short wave
   ln_rnf      = .true.    !  runoffs                                  (T => fill namsbc_rnf)
   ln_ssr      = .true.    !  Sea Surface Restoring on T and/or S      (T => fill namsbc_ssr)
   nn_fwb      = 3         !  FreshWater Budget: =0 unchecked
                           !     =1 global mean of e-p-r set to zero at each time step
                           !     =2 annual global mean of e-p-r set to zero
                           !     =3 global emp set to zero and spread out over erp area
/
```

The ocean needs six fields as surface boundary condition :

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

plus an optional field :

– the atmospheric pressure at the ocean surface (p_a)

Four different ways to provide the first six fields to the ocean are available which are controlled by namelist variables : an analytical formulation (*ln_ana* = true), a flux formulation (*ln_flx* = true), a bulk formulae formulation (CORE (*ln_core* = true) or CLIO (*ln_clio* = true) bulk formulae) and a coupled formulation (exchanges with a atmospheric model via the OASIS coupler) (*ln_cpl* = true). When used, the atmospheric pressure forces both ocean and ice dynamics (*ln_apr_dyn* = true) [1]. The frequency at which the six or seven fields have to be updated is the *nn_fsbc* namelist parameter. When the fields are supplied from data files (flux and bulk formulations), the input fields need not be supplied on the model grid. Instead a file of coordinates and weights can be supplied which maps the data from the supplied grid to the model points (so called "Interpolation on the Fly", see §7.2.2). In addition, the resulting fields can be further modified using several namelist options. These options control the rotation of vector components supplied relative to an east-north coordinate system onto the local grid directions in the model ; the addition of a surface restoring term to observed SST and/or SSS (*ln_ssr* = true) ; the modification of fluxes below ice-covered areas (using observed ice-cover or a sea-ice model) (*nn_ice* = 0,1, 2 or 3) ; the addition of river runoffs as surface freshwater fluxes or lateral inflow (*ln_rnf* = true) ; the addition of a freshwater flux adjustment in order to avoid a mean sea-level drift (*nn_fwb* = 0, 1 or 2) ; and the transformation of the solar radiation (if provided as daily mean) into a diurnal cycle (*ln_dm2dc* = true).

[1] The surface pressure field could be use in bulk formulae, nevertheless none of the current bulk formulea (CLIO and CORE) uses the it.

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, followed by the description of the atmospheric pressure and the river runoff. Next the scheme for interpolation on the fly is described. Finally, the different options that further modify the fluxes applied to the ocean are discussed.

7.1 Surface boundary condition for the ocean

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

$$\left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \Bigg|_{z=1} = \frac{1}{\rho_o} \begin{pmatrix} \tau_u \\ \tau_v \end{pmatrix} \tag{7.1}$$

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

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

$$\frac{\partial T}{\partial t} \equiv \cdots + \left. \frac{Q_{ns}}{\rho_o \, C_p \, e_{3t}} \right|_{k=1} \tag{7.2}$$

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

$$\frac{\partial T}{\partial t} \equiv \cdots + \frac{Q_{sr}}{\rho_o C_p \, e_{3t}} \delta_k \left[I_w \right] \tag{7.3}$$

where I_w is a non-dimensional function that describes the way the light penetrates inside the water column. It is generally a sum of decreasing exponentials (see §5.4.2).

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

$$\begin{aligned} \frac{\partial \eta}{\partial t} &\equiv \cdots + emp \\ \frac{\partial S}{\partial t} &\equiv \cdots + \left. \frac{emp_S \, S}{e_{3t}} \right|_{k=1} \end{aligned} \tag{7.4}$$

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

TAB. 7.1 – Ocean variables provided by the ocean to the surface module (SBC). The variable are averaged over nf_sbc time step, $i.e.$ the frequency of computation of surface fluxes.

In the real ocean, *emp* = emp_S and the ocean salt content is conserved, but it exist several numerical reasons why this equality should be broken. For example, when the ocean is coupled to a sea-ice model, the water exchanged between ice and ocean is slightly salty (mean sea-ice salinity is $\sim$*4 psu*). In this case, emp_S take into account both concentration/dilution effect associated with freezing/melting and the salt flux between ice and ocean, while *emp* is only the volume flux. In addition, in the current version of *NEMO* , the sea-ice is assumed to be above the ocean (the so-called levitating sea-ice). Freezing/melting does not change the ocean volume (no impact on *emp*) but it modifies the SSS.

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

The ocean model provides the surface currents, temperature and salinity averaged over *nf_sbc* time-step (7.1).The computation of the mean is done in *sbcmod.F90* module.

7.2 Input Data generic interface

A generic interface has been introduced to manage the way input data (2D or 3D fields, like surface forcing or ocean T and S) are specify in *NEMO* . This task is archieved by fldread.F90. The module was design with four main objectives in mind :

1. optionally provide a time interpolation of the input data at model time-step, whatever their input frequency is, and according to the different calendars available in the model.
2. optionally provide an on-the-fly space interpolation from the native input data grid to the model grid.
3. make the run duration independent from the period cover by the input files.

4. provide a simple user interface and a rather simple developer interface by limiting the number of prerequisite information.

As a results the user have only to fill in for each variable a structure in the namelist file to defined the input data file and variable names, the frequency of the data (in hours or months), whether its is climatological data or not, the period covered by the input file (one year, month, week or day), and two additional parameters for on-the-fly interpolation. When adding a new input variable, the developer has to add the associated structure in the namelist, read this information by mirroring the namelist read in *sbc_blk_init* for example, and simply call *fld_read* to obtain the desired input field at the model time-step and grid points.

The only constraints are that the input file is a NetCDF file, the file name follows a nomenclature (see §7.2.1), the period it cover is one year, month, week or day, and, if on-the-fly interpolation is used, a file of weights must be supplied (see §7.2.2).

Note that when an input data is archived on a disc which is accessible directly from the workspace where the code is executed, then the use can set the *cn_dir* to the pathway leading to the data. By default, the data are assumed to have been copied so that cn_dir='./'.

7.2.1 Input Data specification (*fldread.F90*)

The structure associated with an input variable contains the following information :

```
!  file name  ! frequency (hours) ! variable  ! time interp. !  clim  ! 'yearly'/ ! weights  ! rotation !
!             !  (if <0  months)  !   name    !   (logical)  !  (T/F) ! 'monthly' ! filename ! pairing  !
```

where

File name : the stem name of the NetCDF file to be open. This stem will be completed automatically by the model, with the addition of a '.nc' at its end and by date information and possibly a prefix (when using AGRIF). Tab.7.2 provides the resulting file name in all possible cases according to whether it is a climatological file or not, and to the open/close frequency (see below for definition).

Record frequency : the frequency of the records contained in the input file. Its unit is in hours if it is positive (for example 24 for daily forcing) or in months if negative (for example -1 for monthly forcing or -12 for annual forcing). Note that this frequency must really be an integer and not a real. On some computers, seting it to '24.' can be interpreted as 240 !

Variable name : the name of the variable to be read in the input NetCDF file.

Time interpolation : a logical to activate, or not, the time interpolation. If set to 'false', the forcing will have a steplike shape remaining constant during each forcing period. For example, when using a daily forcing without time interpolation, the forcing remaining constant from 00h00'00" to 23h59'59". If set to 'true', the forcing will have a broken line shape. Records are assumed to be dated the middle of the

	daily or weekLLL	monthly	yearly
clim = false	fn_yYYYYmMMdDD	fn_yYYYYmMM	fn_yYYYY
clim = true	not possible	fn_m ? ?.nc	fn

TAB. 7.2 – naming nomenclature for climatological or interannual input file, as a function of the Open/close frequency. The stem name is assumed to be 'fn'. For weekly files, the 'LLL' corresponds to the first three letters of the first day of the week (*i.e.* 'sun','sat','fri','thu','wed','tue','mon'). The 'YYYY', 'MM' and 'DD' should be replaced by the actual year/month/day, always coded with 4 or 2 digits. Note that (1) in mpp, if the file is split over each subdomain, the suffix '.nc' is replaced by '_PPPP.nc', where 'PPPP' is the process number coded with 4 digits ; (2) when using AGRIF, the prefix ÔN_Õ is added to files, where 'N' is the child grid number.

forcing period. For example, when using a daily forcing with time interpolation, linear interpolation will be performed between mid-day of two consecutive days.

Climatological forcing : a logical to specify if a input file contains climatological forcing which can be cycle in time, or an interannual forcing which will requires additional files if the period covered by the simulation exceed the one of the file. See the above the file naming strategy which impacts the expected name of the file to be opened.

Open/close frequency : the frequency at which forcing files must be opened/closed. Four cases are coded : 'daily', 'weekLLL' (with 'LLL' the first 3 letters of the first day of the week), 'monthly' and 'yearly' which means the forcing files will contain data for one day, one week, one month or one year. Files are assumed to contain data from the beginning of the open/close period. For example, the first record of a yearly file containing daily data is Jan 1st even if the experiment is not starting at the beginning of the year.

Others : 'weights filename' and 'pairing rotation' are associted with on-the-fly interpolation which is described in §7.2.2.

Additional remarks :

(1) The time interpolation is a simple linear interpolation between two consecutive records of the input data. The only tricky point is therefore to specify the date at which we need to do the interpolation and the date of the records read in the input files. Following **?**, the date of a time step is set at the middle of the time step. For example, for an experiment starting at 0h00'00" with a one hour time-step, a time interpolation will be performed at the following time : 0h30'00", 1h30'00", 2h30'00", etc. However, for forcing data related to the surface module, values are not needed at every time-step but at every *nn_fsbc* time-step. For example with *nn_fsbc* = 3, the surface module will be called at time-steps 1, 4, 7, etc. The date used for the time interpolation is thus redefined to be at the middle of *nn_fsbc* time-step period. In the previous example, this leads to : 1h30'00", 4h30'00", 7h30'00", etc.

(2) For code readablility and maintenance issues, we don't take into account the NetCDF input file calendar. The calendar associated with the forcing field is build according to the information provided by user in the record frequency, the open/close frequency and the type of temporal interpolation. For example, the first record of a yearly file containing daily data that will be interpolated in time is assumed to be start Jan 1st at 12h00'00" and end Dec 31st at 12h00'00".
(3) If a time interpolation is requested, the code will pick up the needed data in the previous (next) file when interpolating data with the first (last) record of the open/close period. For example, if the input file specifications are "yearly, containing daily data to be interpolated in time", the values given by the code between 00h00'00" and 11h59'59" on Jan 1st will be interpolated values between Dec 31st 12h00'00" and Jan 1st 12h00'00". If the forcing is climatological, Dec and Jan will be keep-up from the same year. However, if the forcing is not climatological, at the end of the open/close period the code will automatically close the current file and open the next one. Note that, if the experiment is starting (ending) at the beginning (end) of an open/close period we do accept that the previous (next) file is not existing. In this case, the time interpolation will be performed between two identical values. For example, when starting an experiment on Jan 1st of year Y with yearly files and daily data to be interpolated, we do accept that the file related to year Y-1 is not existing. The value of Jan 1st will be used as the missing one for Dec 31st of year Y-1. If the file of year Y-1 exists, the code will read its last record. Therefore, this file can contain only one record corresponding to Dec 31st, a useful feature for user considering that it is too heavy to manipulate the complete file for year Y-1.

7.2.2 Interpolation on-the-Fly

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

Bilinear Interpolation

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

on the model grid, so that values src01(i,j) and wgt01(i,j) are used to generate a value for point (i,j) in the model.

Symbolically, the algorithm used is :

$$f_m(i,j) = f_m(i,j) + \sum_{k=1}^{4} wgt(k) f(idx(src(k))) \tag{7.5}$$

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

Bicubic Interpolation

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

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

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

Implementation

To activate this option, a non-empty string should be supplied in the weights filename column of the relevant namelist ; if this is left as an empty string no action is taken. In the model, weights files are read in and stored in a structured type (WGT) in the fldread module, as and when they are first required. This initialisation procedure determines whether the input data grid should be treated as cyclical or not by inspecting a global attribute stored in the weights input file. This attribute must be called "ew_wrap" and be of integer type. If it is negative, the input non-model grid is assumed not to be cyclic. If zero or greater, then the value represents the number of columns that overlap. $E.g.$ if the input grid has columns at longitudes 0, 1, 2, ... , 359, then ew_wrap should be set to 0 ; if longitudes are 0.5, 2.5, ... , 358.5, 360.5, 362.5, ew_wrap should be 2. If the model does not find attribute ew_wrap, then a value of -999 is assumed. In this case the *fld_read* routine defaults ew_wrap to value 0 and therefore the grid is assumed to be cyclic with no overlapping columns. (In fact this only matters when bicubic interpolation is required.) Note that no testing is done to check the validity in the model, since there is no way of knowing the name used for the longitude variable, so it is up to the user to make sure his or her data is correctly represented.

Next the routine reads in the weights. Bicubic interpolation is assumed if it finds a variable with name "src05", otherwise bilinear interpolation is used. The WGT structure

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

Limitations

1. The case where input data grids are not logically rectangular has not been tested.
2. This code is not guaranteed to produce positive definite answers from positive definite inputs when a bicubic interpolation method is used.
3. The cyclic condition is only applied on left and right columns, and not to top and bottom rows.
4. The gradients across the ends of a cyclical grid assume that the grid spacing between the two columns involved are consistent with the weights used.
5. Neither interpolation scheme is conservative. (There is a conservative scheme available in SCRIP, but this has not been implemented.)

Utilities

A set of utilities to create a weights file for a rectilinear input grid is available (see the directory NEMOGCM/TOOLS/WEIGHTS).

7.3 Analytical formulation (*sbcana.F90* module)

```
!-----------------------------------------------------------------------
&namsbc_ana    !   analytical surface boundary condition
!-----------------------------------------------------------------------
   nn_tau000   =   0       !  gently increase the stress over the first ntau_rst time-steps
   rn_utau0    =   0.5     !  uniform value for the i-stress
   rn_vtau0    =   0.e0    !  uniform value for the j-stress
   rn_qns0     =   0.e0    !  uniform value for the total heat flux
   rn_qsr0     =   0.e0    !  uniform value for the solar radiation
   rn_emp0     =   0.e0    !  uniform value for the freswater budget (E-P)
/
```

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

If a user wants to apply a different analytical forcing, the *sbcana.F90* module can be modified to use another scheme. As an example, the *sbc_ana_gyre.F90* routine provides

the analytical forcing for the GYRE configuration (see GYRE configuration manual, in preparation).

7.4 Flux formulation (*sbcflx.F90* module)

```
!-----------------------------------------------------------------------
&namsbc_flx    !   surface boundary condition : flux formulation
!-----------------------------------------------------------------------
!             !  file name  ! frequency (hours) ! variable  ! time interp. !  clim  ! 'yearly'/ ! weights  ! rotation !
!             !             !  (if <0  months)  !   name    !   (logical)  !  (T/F) ! 'monthly' ! filename ! pairing  !
   sn_utau     = 'utau'      ,        24         , 'utau'    , .false.      , .false., 'yearly'  , ''       , ''
   sn_vtau     = 'vtau'      ,        24         , 'vtau'    , .false.      , .false., 'yearly'  , ''       , ''
   sn_qtot     = 'qtot'      ,        24         , 'qtot'    , .false.      , .false., 'yearly'  , ''       , ''
   sn_qsr      = 'qsr'       ,        24         , 'qsr'     , .false.      , .false., 'yearly'  , ''       , ''
   sn_emp      = 'emp'       ,        24         , 'emp'     , .false.      , .false., 'yearly'  , ''       , ''

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

In the flux formulation (*ln_flx*=true), the surface boundary condition fields are directly read from input files. The user has to define in the namelist namsbc_flx the name of the file, the name of the variable read in the file, the time frequency at which it is given (in hours), and a logical setting whether a time interpolation to the model time step is required for this field. See §7.2.1 for a more detailed description of the parameters.

Note that in general, a flux formulation is used in associated with a restoring term to observed SST and/or SSS. See §7.9.3 for its specification.

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

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

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

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

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

```
!-----------------------------------------------------------------------
&namsbc_core   !   namsbc_core  CORE bulk formulea
!-----------------------------------------------------------------------
!             !  file name  ! frequency (hours) ! variable  ! time interp. !  clim  ! 'yearly'/ ! weights  ! rotation !
!             !             !  (if <0  months)  !   name    !   (logical)  !  (T/F) ! 'monthly' ! filename ! pairing  !
   sn_wndi = 'u_10.15JUNE2009_orca2'       ,  6  , 'U_10_MOD',   .false.    , .true. , 'yearly'  , ''       , 'Uwnd'
   sn_wndj = 'v_10.15JUNE2009_orca2'       ,  6  , 'V_10_MOD',   .false.    , .true. , 'yearly'  , ''       , 'Vwnd'
   sn_qsr  = 'ncar_rad.15JUNE2009_orca2'   , 24  , 'SWDN_MOD',   .false.    , .true. , 'yearly'  , ''       , ''
   sn_qlw  = 'ncar_rad.15JUNE2009_orca2'   , 24  , 'LWDN_MOD',   .false.    , .true. , 'yearly'  , ''       , ''
   sn_tair = 't_10.15JUNE2009_orca2'       ,  6  , 'T_10_MOD',   .false.    , .true. , 'yearly'  , ''       , ''
   sn_humi = 'q_10.15JUNE2009_orca2'       ,  6  , 'Q_10_MOD',   .false.    , .true. , 'yearly'  , ''       , ''
   sn_prec = 'ncar_precip.15JUNE2009_orca2', -1  , 'PRC_MOD1',   .false.    , .true. , 'yearly'  , ''       , ''
   sn_snow = 'ncar_precip.15JUNE2009_orca2', -1  , 'SNOW'    ,   .false.    , .true. , 'yearly'  , ''       , ''
   sn_tdif = 'taudif_core'                 , 24  , 'taudif'  ,   .false.    , .true. , 'yearly'  , ''       , ''

   cn_dir      = './'      !  root directory for the location of the bulk files
   ln_2m       = .false.   !  air temperature and humidity referenced at 2m (T) instead 10m (F)
   ln_taudif   = .false.   !  HF tau contribution: use "mean of stress module - module of the mean stress" data
   rn_pfac     = 1.        !  multiplicative factor for precipitation (total & snow)
/
```

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

Note that substituting ERA40 to NCEP reanalysis fields does not require changes in the bulk formulea themself. This is the so-called DRAKKAR Forcing Set (DFS) [**?**].

The required 8 input fields are :

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

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

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

```
!-----------------------------------------------------------------------
&namsbc_clio   !   namsbc_clio  CLIO bulk formulea
!-----------------------------------------------------------------------
!             !  file name  ! frequency (hours) ! variable  ! time interp. !  clim  ! 'yearly'/ ! weights  ! rotation !
!             !             !  (if <0  months)  !   name    !   (logical)  !  (T/F) ! 'monthly' ! filename ! pairing  !
   sn_utau     = 'taux_1m'   ,       -1          , 'sozotaux',   .true.     , .true. , 'yearly'  , ''       , ''
   sn_vtau     = 'tauy_1m'   ,       -1          , 'sometauy',   .true.     , .true. , 'yearly'  , ''       , ''
   sn_wndm     = 'flx'       ,       -1          , 'socliowi',   .true.     , .true. , 'yearly'  , ''       , ''
   sn_tair     = 'flx'       ,       -1          , 'socliot2',   .true.     , .true. , 'yearly'  , ''       , ''
   sn_humi     = 'flx'       ,       -1          , 'socliohu',   .true.     , .true. , 'yearly'  , ''       , ''
   sn_ccov     = 'flx'       ,       -1          , 'socliocl',   .false.    , .true. , 'yearly'  , ''       , ''
   sn_prec     = 'flx'       ,       -1          , 'socliopl',   .false.    , .true. , 'yearly'  , ''       , ''

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

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

The required 7 input fields are :

As for the flux formulation, information about the input data required by the model is provided in the namsbc_blk_core or namsbc_blk_clio namelist (see §7.2.1).

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

7.6 Coupled formulation (*sbccpl.F90* module)

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

In the coupled formulation of the surface boundary condition, the fluxes are provided by the OASIS coupler at a frequency which is defined in the OASIS coupler, while sea and ice surface temperature, ocean and ice albedo, and ocean currents are sent to the atmospheric component.

A generalised coupled interface has been developed. It is currently interfaced with OASIS 3 (**key_oasis3**) and does not support OASIS 4 [2]. It has been successfully used to interface *NEMO* to most of the European atmospheric GCM (ARPEGE, ECHAM, ECMWF, HadAM, LMDz), as well as to WRF (Weather Research and Forecasting Model).

Note that in addition to the setting of *ln_cpl* to true, the **key_coupled** have to be defined. The CPP key is mainly used in sea-ice to ensure that the atmospheric fluxes are actually recieved by the ice-ocean system (no calculation of ice sublimation in coupled mode). When PISCES biogeochemical model (**key_top** and **key_pisces**) is also used in the coupled system, the whole carbon cycle is computed by defining **key_cpl_carbon_cycle**. In this case, CO_2 fluxes are exchanged between the atmosphere and the ice-ocean system.

[2] The **key_oasis4** exist. It activates portion of the code that are still under development.

7.7 Atmospheric pressure (*sbcapr.F90*)

```
!-----------------------------------------------------------------------
&namsbc_apr    !   Atmospheric pressure used as ocean forcing or in bulk
!-----------------------------------------------------------------------
!              ! file name ! frequency (hours) ! variable ! time interpol. !  clim   ! 'yearly'/ ! weights  ! rotation !
!              !          !  (if <0  months)  !   name   !    (logical)   !  (T/F)  ! 'monthly' ! filename ! pairing  !
   sn_apr      = 'patm'    ,         -1        ,'somslpre',    .true.      , .true.  , 'yearly'  , ''      ,   ''

   cn_dir      = './'      !  root directory for the location of the bulk files
   ln_ref_apr  = .false.   !  ref. pressure: global mean Patm (T) or a constant (F)
/
```

The optional atmospheric pressure can be used to force ocean and ice dynamics (*ln_apr_dyn* = true, *namsbc* namelist). The input atmospheric forcing defined via *sn_apr* structure (*namsbc_apr* namelist) can be interpolated in time to the model time step, and even in space when the interpolation on-the-fly is used. When used to force the dynamics, the atmospheric pressure is further transformed into an equivalent inverse barometer sea surface height, η_{ib}, using :

$$\eta_{ib} = -\frac{1}{g\,\rho_o}\,(P_{atm} - P_o) \tag{7.6}$$

where P_{atm} is the atmospheric pressure and P_o a reference atmospheric pressure. A value of $101,000\ N/m^2$ is used unless *ln_ref_apr* is set to true. In this case P_o is set to the value of P_{atm} averaged over the ocean domain, $i.e.$ the mean value of η_{ib} is kept to zero at all time step.

The gradient of η_{ib} is added to the RHS of the ocean momentum equation (see *dynspg.F90* for the ocean). For sea-ice, the sea surface height, η_m, which is provided to the sea ice model is set to $\eta - \eta_{ib}$ (see *sbcssr.F90* module). η_{ib} can be set in the output. This can simplify altimetry data and model comparison as inverse barometer sea surface height is usually removed from these date prior to their distribution.

7.8 River runoffs (*sbcrnf.F90*)

```
!-----------------------------------------------------------------------
&namsbc_rnf    !   runoffs namelist surface boundary condition
!-----------------------------------------------------------------------
!              !  file name  ! frequency (hours) ! variable  ! time interp. !  clim  ! 'yearly'/ ! weights  ! rotation !
!              !             !  (if <0  months)  !   name    !   (logical)  !  (T/F) ! 'monthly' ! filename ! pairing  !
   sn_rnf      = 'runoff_core_monthly',    -1    , 'sorunoff',   .true.     , .true. , 'yearly'  , ''       , ''
   sn_cnf      = 'runoff_core_monthly',     0    , 'socoefr0',   .false.    , .true. , 'yearly'  , ''       , ''
   sn_s_rnf    = 'runoffs'            ,    24    , 'rosaline',   .true.     , .true. , 'yearly'  , ''       , ''
   sn_t_rnf    = 'runoffs'            ,    24    , 'rotemper',   .true.     , .true. , 'yearly'  , ''       , ''
   sn_dep_rnf  = 'runoffs'            ,     0    , 'rodepth' ,   .false.    , .true. , 'yearly'  , ''       , ''

   cn_dir       = './'      !  root directory for the location of the runoff files
   ln_rnf_emp   = .false.   !  runoffs included into precipitation field (T) or into a file (F)
   ln_rnf_mouth = .true.    !  specific treatment at rivers mouths
   rn_hrnf      =  15.e0    !  depth over which enhanced vertical mixing is used
   rn_avt_rnf   =   1.e-3   !  value of the additional vertical mixing coef. [m2/s]
   rn_rfact     =   1.e0    !  multiplicative factor for runoff
   ln_rnf_depth = .false.   !  read in depth information for runoff
   ln_rnf_tem   = .false.   !  read in temperature information for runoff
   ln_rnf_sal   = .false.   !  read in salinity information for runoff
/
```

River runoff generally enters the ocean at a nonzero depth rather than through the surface. Many models, however, have traditionally inserted river runoff to the top model cell. This was the case in *NEMO* prior to the version 3.3, and was combined with an option to increase vertical mixing near the river mouth.

However, with this method numerical and physical problems arise when the top grid cells are of the order of one meter. This situation is common in coastal modelling and is becoming more common in open ocean and climate modelling [3].

As such from V 3.3 onwards it is possible to add river runoff through a non-zero depth, and for the temperature and salinity of the river to effect the surrounding ocean. The user is able to specify, in a NetCDF input file, the temperature and salinity of the river, along with the depth (in metres) which the river should be added to.

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

The runoff value and attributes are read in in sbcrnf. For temperature -999 is taken as missing data and the river temperature is taken to be the surface temperatue at the river point. For the depth parameter a value of -1 means the river is added to the surface box only, and a value of -999 means the river is added through the entire water column. After being read in the temperature and salinity variables are multiplied by the amount of runoff (converted into m/s) to give the heat and salt content of the river runoff. After the user specified depth is read ini, the number of grid boxes this corresponds to is calculated and stored in the variable *nz_rnf*. The variable *h_dep* is then calculated to be the depth (in metres) of the bottom of the lowest box the river water is being added to (i.e. the total depth that river water is being added to in the model).

The mass/volume addition due to the river runoff is, at each relevant depth level, added to the horizontal divergence (*hdivn*) in the subroutine *sbc_rnf_div* (called from *divcur.F90*). This increases the diffusion term in the vicinity of the river, thereby simulating a momentum flux. The sea surface height is calculated using the sum of the horizontal divergence terms, and so the river runoff indirectly forces an increase in sea surface height.

The *hdivn* terms are used in the tracer advection modules to force vertical velocities. This causes a mass of water, equal to the amount of runoff, to be moved into the box above. The heat and salt content of the river runoff is not included in this step, and so the tracer concentrations are diluted as water of ocean temperature and salinity is moved upward out of the box and replaced by the same volume of river water with no corresponding heat and salt addition.

For the linear free surface case, at the surface box the tracer advection causes a flux of water (of equal volume to the runoff) through the sea surface out of the domain, which causes a salt and heat flux out of the model. As such the volume of water does not change, but the water is diluted.

For the non-linear free surface case (**key_vvl**), no flux is allowed through the surface. Instead in the surface box (as well as water moving up from the boxes below) a volume of runoff water is added with no corresponding heat and salt addition and so as happens in the lower boxes there is a dilution effect. (The runoff addition to the top box along with the water being moved up through boxes below means the surface box has a large increase

[3] At least a top cells thickness of 1 meter and a 3 hours forcing frequency are required to properly represent the diurnal cycle [**?**]. see also §7.9.1.

in volume, whilst all other boxes remain the same size)

In trasbc the addition of heat and salt due to the river runoff is added. This is done in the same way for both vvl and non-vvl. The temperature and salinity are increased through the specified depth according to the heat and salt content of the river.

In the non-linear free surface case (vvl), near the end of the time step the change in sea surface height is redistrubuted through the grid boxes, so that the original ratios of grid box heights are restored. In doing this water is moved into boxes below, throughout the water column, so the large volume addition to the surface box is spread between all the grid boxes.

It is also possible for runnoff to be specified as a negative value for modelling flow through straits, i.e. modelling the Baltic flow in and out of the North Sea. When the flow is out of the domain there is no change in temperature and salinity, regardless of the namelist options used, as the ocean water leaving the domain removes heat and salt (at the same concentration) with it.

7.9 Miscellaneous options

7.9.1 Diurnal cycle (*sbcdcy.F90*)

? have shown that to capture 90% of the diurnal variability of SST requires a vertical resolution in upper ocean of 1 m or better and a temporal resolution of the surface fluxes of 3 h or less. Unfortunately high frequency forcing fields are rare, not to say inexistent. Nevertheless, it is possible to obtain a reasonable diurnal cycle of the SST knowning only short wave flux (SWF) at high frequency [**?**]. Furthermore, only the knowledge of daily mean value of SWF is needed, as higher frequency variations can be reconstructed from them, assuming that the diurnal cycle of SWF is a scaling of the top of the atmosphere diurnal cycle of incident SWF. The **?** reconstruction algorithm is available in *NEMO* by setting *ln_dm2dc* = true (a *namsbc* namelist parameter) when using CORE bulk formulea (*ln_blk_core* = true) or the flux formulation (*ln_flx* = true). The reconstruction is performed in the *sbcdcy.F90* module. The detail of the algoritm used can be found in the appendix A of **?**. The algorithm preserve the daily mean incomming SWF as the reconstructed SWF at a given time step is the mean value of the analytical cycle over this time step (Fig.7.1). The use of diurnal cycle reconstruction requires the input SWF to be daily ($i.e.$ a frequency of 24 and a time interpolation set to true in *sn_qsr* namelist parameter). Furthermore, it is recommended to have a least 8 surface module time step per day, that is Δt *nn_fsbc* $< 10,800\ s = 3\ h$. An example of recontructed SWF is given in Fig.7.2 for a 12 reconstructed diurnal cycle, one every 2 hours (from 1am to 11pm).

Note also that the setting a diurnal cycle in SWF is highly recommended when the top layer thickness approach 1 m or less, otherwise large error in SST can appear due to an inconsistency between the scale of the vertical resolution and the forcing acting on that scale.

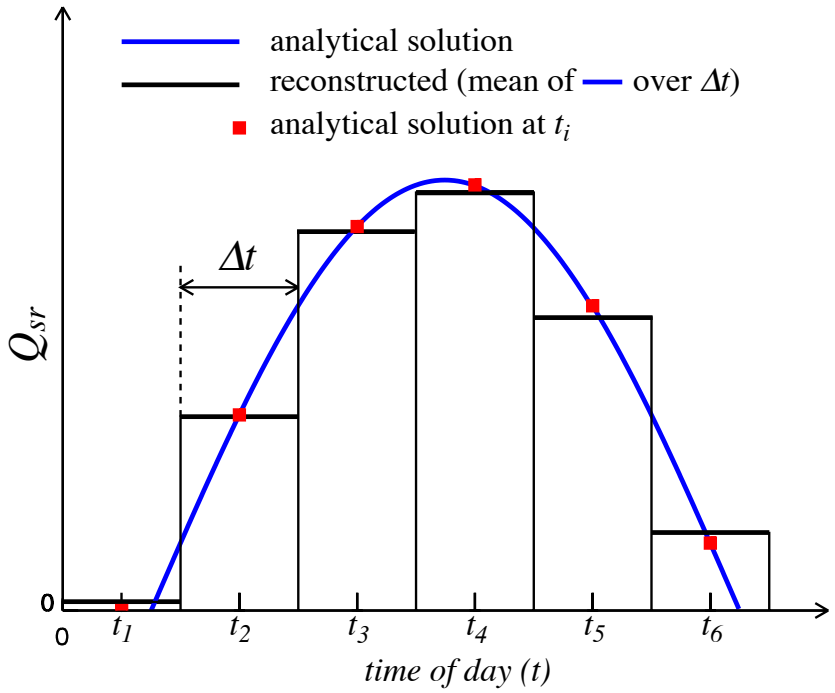

FIG. 7.1 – Example of recontruction of the diurnal cycle variation of short wave flux from daily mean values. The reconstructed diurnal cycle (black line) is chosen as the mean value of the analytical cycle (blue line) over a time step, not as the mid time step value of the analytically cycle (red square). From **?**.

7.9.2 Rotation of vector pairs onto the model grid directions

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

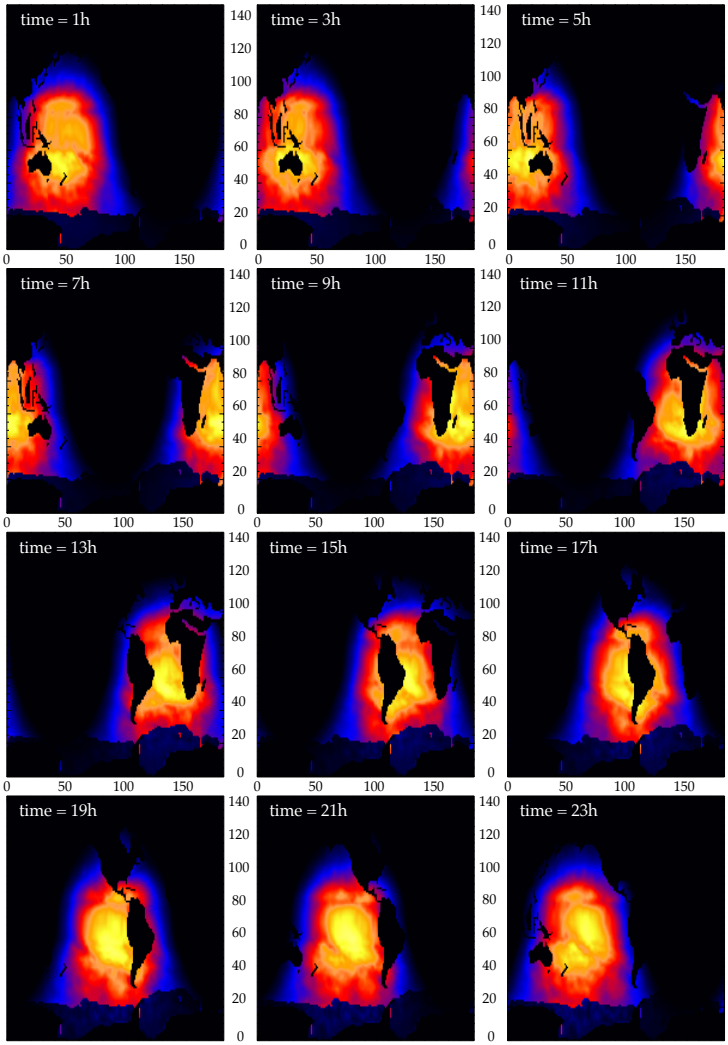

FIG. 7.2 – Example of recontruction of the diurnal cycle variation of short wave flux from daily mean values on an ORCA2 grid with a time sampling of 2 hours (from 1am to 11pm). The display is on (i,j) plane.

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

```
!-----------------------------------------------------------------------
&namsbc_ssr    !   surface boundary condition : sea surface restoring
!-----------------------------------------------------------------------
!             !  file name  ! frequency (hours) ! variable  ! time interp. !  clim  ! 'yearly'/ ! weights  ! rotation !
!             !             !  (if <0  months)  !   name    !   (logical)  !  (T/F) ! 'monthly' ! filename ! pairing  !
   sn_sst     = 'sst_data'  ,        24         ,  'sst'    ,    .false.   , .false., 'yearly'  , ''       , ''
   sn_sss     = 'sss_data'  ,        -1         ,  'sss'    ,    .true.    , .true. , 'yearly'  , ''       , ''

   cn_dir     = './'      !  root directory for the location of the runoff files
   nn_sstr    =     0     !  add a retroaction term in the surface heat       flux (=1) or not (=0)
   nn_sssr    =     2     !  add a damping     term in the surface freshwater flux (=2)
                          !  or to SSS only (=1) or no damping term (=0)
```

```
   rn_dqdt     =   -40.    !  magnitude of the retroaction on temperature   [W/m2/K]
   rn_deds     =   -27.7   !  magnitude of the damping on salinity   [mm/day]
   ln_sssr_bnd =   .true.  !  flag to bound erp term (associated with nn_sssr=2)
   rn_sssr_bnd =   4.e0    !  ABS(Max/Min) value of the damping erp term [mm/day]
/
```

In forced mode using a flux formulation (*ln_flx* = true), a feedback term *must* be added to the surface heat flux Q^o_{ns} :

$$Q_{ns} = Q^o_{ns} + \frac{dQ}{dT}\left(T|_{k=1} - SST_{Obs}\right) \tag{7.7}$$

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

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

$$emp = emp_o + \gamma_s^{-1} e_{3t} \frac{\left(S|_{k=1} - SSS_{Obs}\right)}{S|_{k=1}} \tag{7.8}$$

where emp_o is a net surface fresh water flux (observed, climatological or an atmospheric model product), SSS_{Obs} is a sea surface salinity (usually a time interpolation of the monthly mean Polar Hydrographic Climatology [**?**]), $S|_{k=1}$ is the model surface layer salinity and γ_s is a negative feedback coefficient which is provided as a namelist parameter. Unlike heat flux, there is no physical justification for the feedback term in 7.8 as the atmosphere does not care about ocean surface salinity [**?**]. The SSS restoring term should be viewed as a flux correction on freshwater fluxes to reduce the uncertainties we have on the observed freshwater budget.

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

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

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

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

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

7.9.5 Freshwater budget control (*sbcfwb.F90*)

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

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

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

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

8 Lateral Boundary Condition (LBC)

Contents

8.1 Boundary Condition at the Coast (*rn_shlat*)

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

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

$$\frac{A^{lT}}{e_1}\frac{\partial T}{\partial i} \equiv \frac{A_u^{lT}}{e_{1u}}\,\delta_{i+1/2}\left[T\right] \quad mask_u \tag{8.1}$$

(where mask$_u$ is the mask array at a u-point) ensures that the heat flux is zero inside land and at the boundaries, since mask$_u$ is zero at solid boundaries which in this case are defined at u-points (normal velocity u remains zero at the coast) (Fig. 8.1).

For momentum the situation is a bit more complex as two boundary conditions must be provided along the coast (one each for the normal and tangential velocities). The boundary of the ocean in the C-grid is defined by the velocity-faces. For example, at a given T-level, the lateral boundary (a coastline or an intersection with the bottom topography) is made of segments joining f-points, and normal velocity points are located between two $f-$points (Fig. 8.1). The boundary condition on the normal velocity (no flux through solid boundaries) can thus be easily implemented using the mask system. The boundary condition on the tangential velocity requires a more specific treatment. This boundary condition influences the relative vorticity and momentum diffusive trends, and is required in order to compute the vorticity at the coast. Four different types of lateral boundary condition are available, controlled by the value of the *rn_shlat* namelist parameter. (The value of the mask$_f$ array along the coastline is set equal to this parameter.) These are :

free-slip boundary condition (*rn_shlat*=0) : the tangential velocity at the coastline is equal to the offshore velocity, $i.e.$ the normal derivative of the tangential velocity

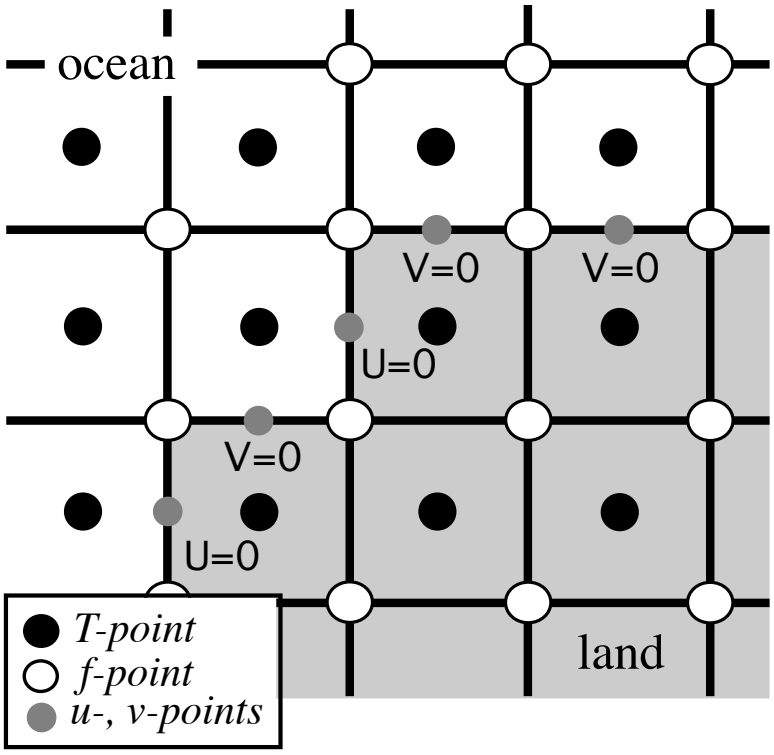

FIG. 8.1 – Lateral boundary (thick line) at T-level. The velocity normal to the boundary is set to zero.

is zero at the coast, so the vorticity : mask_f array is set to zero inside the land and just at the coast (Fig. 8.2-a).

no-slip boundary condition (*rn_shlat*=2) : the tangential velocity vanishes at the coastline. Assuming that the tangential velocity decreases linearly from the closest ocean velocity grid point to the coastline, the normal derivative is evaluated as if the velocities at the closest land velocity gridpoint and the closest ocean velocity gridpoint were of the same magnitude but in the opposite direction (Fig. 8.2-b). Therefore, the vorticity along the coastlines is given by :

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

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

$$\zeta \equiv \frac{1}{e_{1f}\, e_{2f}} \left(\delta_{i+1/2} \left[e_{2v}\, v \right] - \delta_{j+1/2} \left[e_{1u}\, u \right] \right) \ \text{mask}_f \tag{8.2}$$

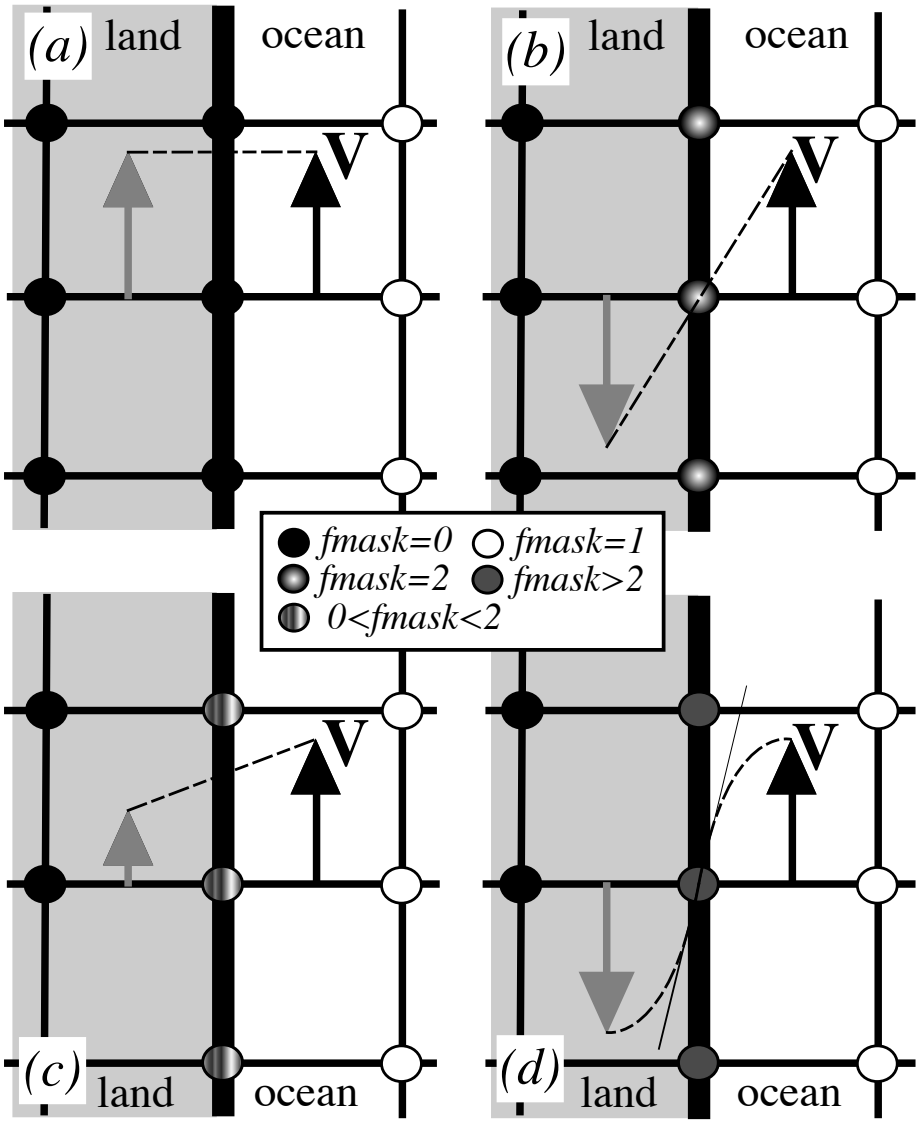

FIG. 8.2 – lateral boundary condition (a) free-slip ($rn\_shlat = 0$); (b) no-slip ($rn\_shlat = 2$); (c) "partial" free-slip ($0 < rn\_shlat < 2$) and (d) "strong" no-slip ($2 < rn\_shlat$). Implied "ghost" velocity inside land area is display in grey.

"partial" free-slip boundary condition (0<*rn_shlat*<2) : the tangential velocity at the coastline is smaller than the offshore velocity, $i.e.$ there is a lateral friction but not strong enough to make the tangential velocity at the coast vanish (Fig. 8.2-c). This can be selected by providing a value of mask$_f$ strictly inbetween 0 and 2.

"strong" no-slip boundary condition (2<*rn_shlat*) : the viscous boundary layer is assumed to be smaller than half the grid size (Fig. 8.2-d). The friction is thus larger than in the no-slip case.

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

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

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

8.2 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.

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

The choice of closed, cyclic or symmetric model domain boundary condition is made by setting *jperio* to 0, 1 or 2 in file *par_oce.F90*. Each time such a boundary condition is needed, it is set by a call to routine *lbclnk.F90*. The computation of momentum and tracer trends proceeds from $i = 2$ to $i = jpi - 1$ and from $j = 2$ to $j = jpj - 1$, $i.e.$ in the model interior. To choose a lateral model boundary condition is to specify the first and last rows and columns of the model variables.

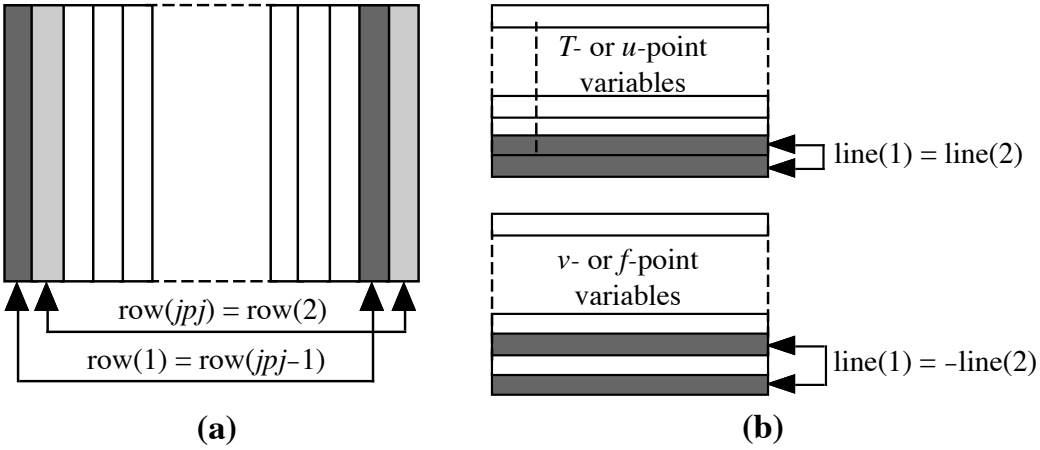

FIG. 8.3 – setting of (a) east-west cyclic (b) symmetric across the equator boundary conditions.

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

For cyclic east-west boundary (*jperio=1*) , first and last rows are set to zero (closed) whilst the first column is set to the value of the last-but-one column and the last column to the value of the second one (Fig. 8.3-a). Whatever flows out of the eastern (western) end of the basin enters the western (eastern) end. Note that there is no option for north-south cyclic or for doubly cyclic cases.

For symmetric boundary condition across the equator (*jperio=2*) , last rows, and first and last columns are set to zero (closed). The row of symmetry is chosen to be the u- and $T-$points equator line ($j = 2$, i.e. at the southern end of the domain). For arrays defined at $u-$ or $T-$points, the first row is set to the value of the third row while for most of v- and f-point arrays (v, ζ, $j\psi$, but scalar arrays such as eddy coefficients) the first row is set to minus the value of the second row (Fig. 8.3-b). Note that this boundary condition is not yet available for the case of a massively parallel computer (**key_mpp** defined).

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

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

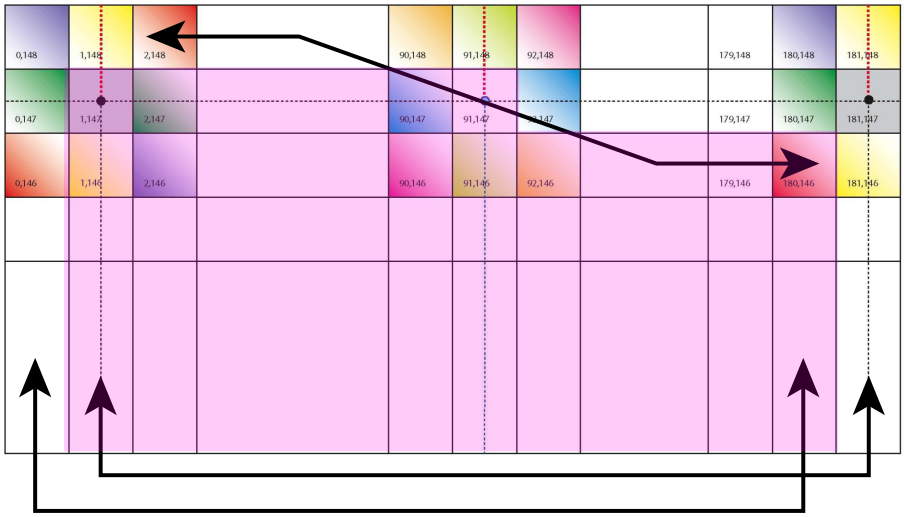

FIG. 8.4 – North fold boundary with a T-point pivot and cyclic east-west boundary condition ($jperio = 4$), as used in ORCA 2, 1/4, and 1/12. Pink shaded area corresponds to the inner domain mask (see text).

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

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

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

The parallelization strategy is defined by the physical characteristics of the ocean model. Second order finite difference schemes lead to local discrete operators that depend

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

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

$$\begin{aligned} jpi &= (jpiglo - 2 * jpreci + (jpni - 1))/jpni + 2 * jpreci \\ jpj &= (jpjglo - 2 * jprecj + (jpnj - 1))/jpnj + 2 * jprecj \end{aligned} \tag{8.3}$$

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

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

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

$$T_g(i + nimpp - 1, j + njmpp - 1, k) = T_l(i, j, k), \tag{8.4}$$

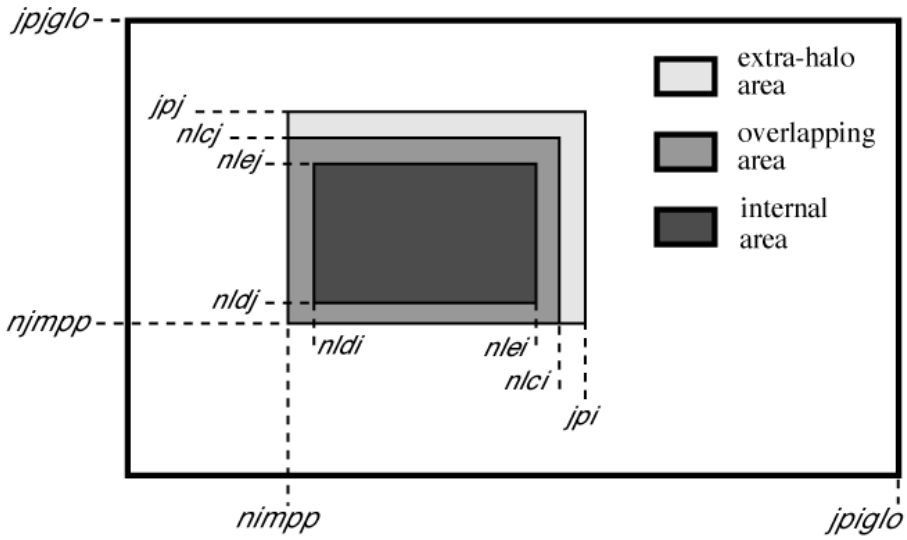

FIG. 8.5 – Positioning of a sub-domain when massively parallel processing is used.

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

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

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

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

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

The *NEMO* model computes equation terms with the help of mask arrays (0 on land points and 1 on sea points). It is easily readable and very efficient in the context of a computer with vectorial architecture. However, in the case of a scalar processor, computations over the land regions become more expensive in terms of CPU time. It is worse when we use a complex configuration with a realistic bathymetry like the global ocean where more than 50 % of points are land points. For this reason, a pre-processing tool can be used to choose the mpp domain decomposition with a maximum number of only land

points processors, which can then be eliminated. (For example, the mpp_optimiz tools, available from the DRAKKAR web site.) This optimisation is dependent on the specific bathymetry employed. The user then chooses optimal parameters *jpni*, *jpnj* and *jpnij* with $jpnij < jpni \times jpnj$, leading to the elimination of $jpni \times jpnj - jpnij$ land processors. When those parameters are specified in module *par_oce.F90*, the algorithm in the *inimpp2* routine sets each processor's parameters (nbound, nono, noea,...) so that the land-only processors are not taken into account.

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

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

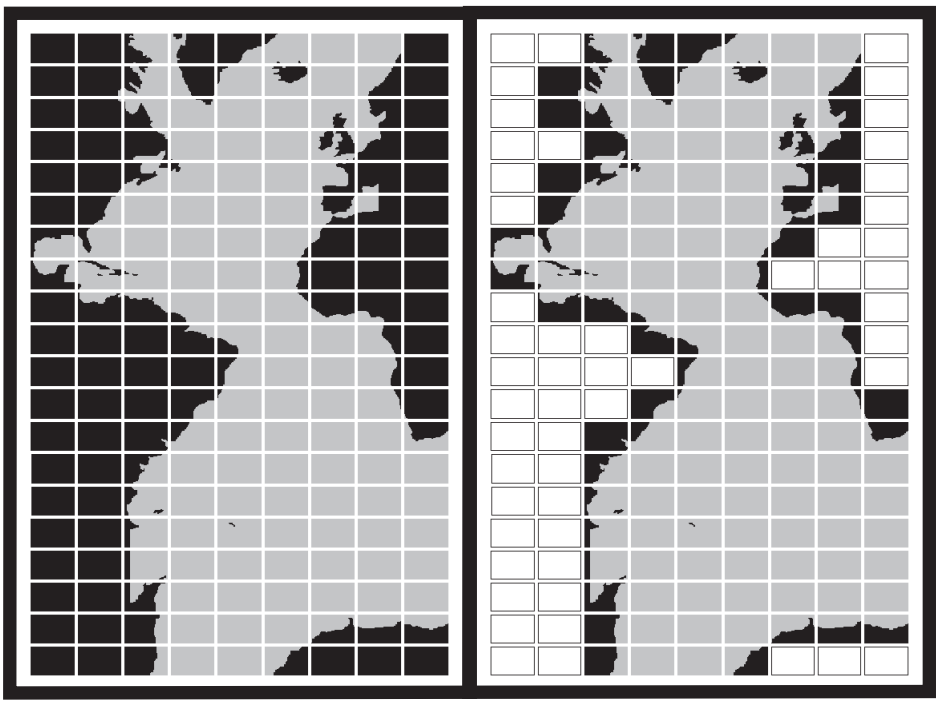

FIG. 8.6 – Example of Atlantic domain defined for the CLIPPER projet. Initial grid is composed of 773 x 1236 horizontal points. (a) the domain is split onto 9 subdomains (jpni=9, jpnj=20). 52 subdomains are land areas. (b) 52 subdomains are eliminated (white rectangles) and the resulting number of processors really used during the computation is jpnij=128.

8.4 Open Boundary Conditions (key_obc) (OBC)

```
!------------------------------------------------------------------------
&namobc        !   open boundaries parameters                          ("key_obc")
!------------------------------------------------------------------------
   ln_obc_clim = .false.   !  climatological obc data files (T) or not (F)
   ln_vol_cst  = .true.    !  impose the total volume conservation (T) or not (F)
   ln_obc_fla  = .false.   !  Flather open boundary condition
   nn_obcdta   =    1      !  = 0 the obc data are equal to the initial state
                           !  = 1 the obc data are read in 'obc.dta' files
   cn_obcdta   = 'annual'  !  set to annual if obc datafile hold 1 year of data
                           !  set to monthly if obc datafile hold 1 month of data
   rn_dpein    =    1.     !  damping time scale for inflow at east  open boundary
   rn_dpwin    =    1.     !     -           -         -       west    -      -
   rn_dpnin    =    1.     !     -           -         -       north   -      -
   rn_dpsin    =    1.     !     -           -         -       south   -      -
   rn_dpeob    = 3000.     !  time relaxation (days) for the east  open boundary
   rn_dpwob    =   15.     !     -           -         -     west    -      -
   rn_dpnob    = 3000.     !     -           -         -     north   -      -
   rn_dpsob    =   15.     !     -           -         -     south   -      -
   rn_volemp   =    1.     !  = 0 the total volume change with the surface flux (E-P-R)
                           !  = 1 the total volume remains constant
/
```

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

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

- tell the model that a boundary is "open" and not closed by a wall, for example by modifying the calculation of the divergence of velocity there ;
- impose values of tracers and velocities at that boundary (values which may be taken from a climatology) : this is the"fixed OBC" option.
- calculate boundary values by a sophisticated algorithm combining radiation and relaxation ("radiative OBC" option)

The package resides in the OBC directory. It is described here in four parts : the boundary geometry (parameters to be set in *obc_par.F90*), the forcing data at the boundaries (module *obcdta.F90*), the radiation algorithm involving the namelist and module *obcrad.F90*, and a brief presentation of boundary update and restart files.

8.4.1 Boundary geometry

First one has to realize that open boundaries may not necessarily be located at the extremities of the computational domain. They may exist in the middle of the domain, for example at Gibraltar Straits if one wants to avoid including the Mediterranean in an Atlantic domain. This flexibility has been found necessary for the CLIPPER project [**?**]. Because of the complexity of the geometry of ocean basins, it may even be necessary to have more than one "west" open boundary, more than one "north", etc. This is not possible with the OBC option : only one open boundary of each kind, west, east, south and north is allowed ; these names refer to the grid geometry (not to the direction of the geographical "west", "east", etc).

The open boundary geometry is set by a series of parameters in the module *obc_par.F90*. For an eastern open boundary, parameters are *lp_obc_east* (true if an east open boundary exists), *jpieob* the i-index along which the eastern open boundary (eob) is located, *jpjed* the j-index at which it starts, and *jpjef* the j-index where it ends (note d is for "début" and f for "fin" in French). Similar parameters exist for the west, south and north cases (Table 8.1).

Boundary and Logical flag	Constant index	Starting index (début)	Ending index (fin)
West lp_obc_west	*jpiwob* $>= 2$ i-index of a u point	*jpjwd*$>= 2$ j of a T point	*jpjwf* ¡= *jpjglo*-1 j of a T point
East lp_obc_east	*jpieob*<=*jpiglo*-2 i-index of a u point	*jpjed* $>= 2$ j of a T point	*jpjef* $<=$ *jpjglo*-1 j of a T point
South lp_obc_south	*jpjsob* $>= 2$ j-index of a v point	*jpisd* $>= 2$ i of a T point	*jpisf*<=*jpiglo*-1 i of a T point
North lp_obc_north	*jpjnob* $<=$ *jpjglo*-2 j-index of a v point	*jpind* $>= 2$ i of a T point	*jpinf*<=*jpiglo*-1 i of a T point

TAB. 8.1 – Names of different indices relating to the open boundaries. In the case of a completely open ocean domain with four ocean boundaries, the parameters take exactly the values indicated.

The open boundaries must be along coordinate lines. On the C-grid, the boundary itself is along a line of normal velocity points : v points for a zonal open boundary (the south or north one), and u points for a meridional open boundary (the west or east one). Another constraint is that there still must be a row of masked points all around the domain, as if the domain were a closed basin (unless periodic conditions are used together with open boundary conditions). Therefore, an open boundary cannot be located at the first/last index, namely, 1, *jpiglo* or *jpjglo*. Also, the open boundary algorithm involves calculating the normal velocity points situated just on the boundary, as well as the tangential velocity and temperature and salinity just outside the boundary. This means that for a west/south boundary, normal velocities and temperature are calculated at the same index *jpiwob* and *jpjsob*, respectively. For an east/north boundary, the normal velocity is calculated at index *jpieob* and *jpjnob*, but the "outside" temperature is at index *jpieob*+1 and *jpjnob*+1. This means that *jpieob*, *jpjnob* cannot be bigger than *jpiglo*-2, *jpjglo*-2.

The starting and ending indices are to be thought of as T point indices : in many cases they indicate the first land T-point, at the extremity of an open boundary (the coast line follows the f grid points, see Fig. 8.7 for an example of a northern open boundary). All indices are relative to the global domain. In the free surface case it is possible to have "ocean corners", that is, an open boundary starting and ending in the ocean.

Although not compulsory, it is highly recommended that the bathymetry in the vicinity of an open boundary follows the following rule : in the direction perpendicular to the

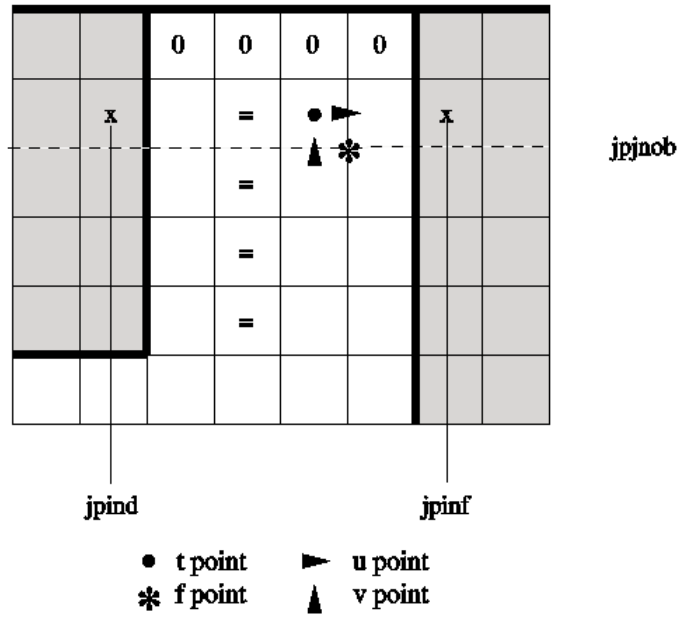

FIG. 8.7 – Localization of the North open boundary points.

open line, the water depth should be constant for 4 grid points. This is in order to ensure that the radiation condition, which involves model variables next to the boundary, is calculated in a consistent way. On Fig.8.7 we indicate by an $=$ symbol, the points which should have the same depth. It means that at the 4 points near the boundary, the bathymetry is cylindrical . The line behind the open T-line must be 0 in the bathymetry file (as shown on Fig.8.7 for example).

8.4.2 Boundary data

It is necessary to provide information at the boundaries. The simplest case is when this information does not change in time and is equal to the initial conditions (namelist variable *nn_obcdta*=0). This is the case for the standard configuration EEL5 with open boundaries. When (*nn_obcdta*=1), open boundary information is read from netcdf files. For convenience the input files are supposed to be similar to the "history" NEMO output files, for dimension names and variable names. Open boundary arrays must be dimensioned according to the parameters of table 8.1 : for example, at the western boundary, arrays have a dimension of *jpwf*-*jpwd*+1 in the horizontal and *jpk* in the vertical.

When ocean observations are used to generate the boundary data (a hydrographic section for example, as in **?**) it happens often that only the velocity normal to the boundary is known, which is the reason why the initial OBC code assumes that only T, S, and the normal velocity (u or v) needs to be specified. As more and more global model solutions

and ocean analysis products become available, it will be possible to provide information about all the variables (including the tangential velocity) so that the specification of four variables at each boundaries will become standard. For the sea surface height, one must distinguish between the filtered free surface case and the time-splitting or explicit treatment of the free surface. In the first case, it is assumed that the user does not wish to represent high frequency motions such as tides. The boundary condition is thus one of zero normal gradient of sea surface height at the open boundaries, following **?**. No information other than the total velocity needs to be provided at the open boundaries in that case. In the other two cases (time splitting or explicit free surface), the user must provide barotropic information (sea surface height and barotropic velocities) and the use of the Flather algorithm for barotropic variables is recommanded. However, this algorithm has not yet been fully tested and bugs remain in NEMO v2.3. Users should read the code carefully before using it. Finally, in the case of the rigid lid approximation the barotropic streamfunction must be provided, as documented in **?**). This option is no longer recommended but remains in NEMO V2.3.

One frequently encountered case is when an open boundary domain is constructed from a global or larger scale NEMO configuration. Assuming the domain corresponds to indices $ib : ie$, $jb : je$ of the global domain, the bathymetry and forcing of the small domain can be created by using the following netcdf utility on the global files : ncks -F $-d\ x, ib, ie\ -d\ y, jb, je$ (part of the nco series of utilities, see their website). The open boundary files can be constructed using ncks commands, following table 8.2.

It is assumed that the open boundary files contain the variables for the period of the model integration. If the boundary files contain one time frame, the boundary data is held fixed in time. If the files contain 12 values, it is assumed that the input is a climatology for a repeated annual cycle (corresponding to the case *ln_obc_clim* =true). The case of an arbitrary number of time frames is not yet implemented correctly ; the user is required to write his own code in the module *obc_dta.F90* to deal with this situation.

8.4.3 Radiation algorithm

The art of open boundary management consists in applying a constraint strong enough that the inner domain "feels" the rest of the ocean, but weak enough that perturbations are allowed to leave the domain with minimum false reflections of energy. The constraints are specified separately at each boundary as time scales for "inflow" and "outflow" as defined below. The time scales are set (in days) by namelist parameters such as *rn_dpein*, *rn_dpeob* for the eastern open boundary for example. When both time scales are zero for a given boundary ($e.g.$ for the western boundary, *lp_obc_west*=true, *rn_dpwob*=0 and *rn_dpwin*=0) this means that the boundary in question is a "fixed " boundary where the solution is set exactly by the boundary data. This is not recommended, except in combination with increased viscosity in a "sponge" layer next to the boundary in order to avoid spurious reflections.

The radiationrelaxation algorithm is applied when either relaxation time (for "inflow" or "outflow") is non-zero. It has been developed and tested in the SPEM model and its suc-

OBC	Variable	file name	Index	Start	end
West	T,S	obcwest_TS.nc	ib+1	jb+1	$je-1$
	U	obcwest_U.nc	ib+1	jb+1	$je-1$
	V	obcwest_V.nc	ib+1	jb+1	$je-1$
East	T,S	obceast_TS.nc	ie-1	jb+1	$je-1$
	U	obceast_U.nc	ie-2	jb+1	$je-1$
	V	obceast_V.nc	ie-1	jb+1	$je-1$
South	T,S	obcsouth_TS.nc	jb+1	ib+1	$ie-1$
	U	obcsouth_U.nc	jb+1	ib+1	$ie-1$
	V	obcsouth_V.nc	jb+1	ib+1	$ie-1$
North	T,S	obcnorth_TS.nc	je-1	ib+1	$ie-1$
	U	obcnorth_U.nc	je-1	ib+1	$ie-1$
	V	obcnorth_V.nc	je-2	ib+1	$ie-1$

TAB. 8.2 – Requirements for creating open boundary files from a global configuration, appropriate for the subdomain of indices $ib : ie$, $jb : je$. "Index" designates the i or j index along which the u of v boundary point is situated in the global configuration, starting and ending with the j or i indices indicated. For example, to generate file obcnorth_V.nc, use the command ncks $-F$ $-d$ $y, je-2$ $-d$ $x, ib+1, ie-1$

cessor ROMS [**??**], which is an s-coordinate model on an Arakawa C-grid. Although the algorithm has been numerically successful in the CLIPPER Atlantic models, the physics do not work as expected [**?**]. Users are invited to consider open boundary conditions (OBC hereafter) with some scepticism [**??**].

The first part of the algorithm calculates a phase velocity to determine whether perturbations tend to propagate toward, or away from, the boundary. Let us consider a model variable ϕ. The phase velocities ($C_{\phi x}$,$C_{\phi y}$) for the variable ϕ, in the directions normal and tangential to the boundary are

$$C_{\phi x} = \frac{-\phi_t}{(\phi_x^2 + \phi_y^2)} \phi_x \qquad C_{\phi y} = \frac{-\phi_t}{(\phi_x^2 + \phi_y^2)} \phi_y. \tag{8.5}$$

Following **?** and **?** we retain only the normal component of the velocity, $C_{\phi x}$, setting $C_{\phi y} = 0$ (but unlike the original Orlanski radiation algorithm we retain ϕ_y in the expression for $C_{\phi x}$).

The discrete form of (8.5), described by **?**, takes into account the two rows of grid points situated inside the domain next to the boundary, and the three previous time steps (n, $n-1$, and $n-2$). The same equation can then be discretized at the boundary at time steps $n-1$, n and $n+1$ in order to extrapolate for the new boundary value ϕ^{n+1}.

In the open boundary algorithm as implemented in NEMO v2.3, the new boundary values are updated differently depending on the sign of $C_{\phi x}$. Let us take an eastern boundary as an example. The solution for variable ϕ at the boundary is given by a generalized wave equation with phase velocity C_ϕ, with the addition of a relaxation term, as :

$$\phi_t = -C_{\phi x}\phi_x + \frac{1}{\tau_o}(\phi_c - \phi) \qquad (C_{\phi x} > 0), \tag{8.6}$$

$$\phi_t = \frac{1}{\tau_i}(\phi_c - \phi) \qquad (C_{\phi x} < 0), \tag{8.7}$$

where ϕ_c is the estimate of ϕ at the boundary, provided as boundary data. Note that in (8.6), $C_{\phi x}$ is bounded by the ratio $\delta x / \delta t$ for stability reasons. When $C_{\phi x}$ is eastward (outward propagation), the radiation condition (8.6) is used. When $C_{\phi x}$ is westward (inward propagation), (8.7) is used with a strong relaxation to climatology (usually τ_i = *rn_dpein* =1 day). Equation (8.7) is solved with a Euler time-stepping scheme. As a consequence, setting τ_i smaller than, or equal to the time step is equivalent to a fixed boundary condition. A time scale of one day is usually a good compromise which guarantees that the inflow conditions remain close to climatology while ensuring numerical stability.

In the case of a western boundary located in the Eastern Atlantic, **?** have been able to implement the radiation algorithm without any boundary data, using persistence from the previous time step instead. This solution has not worked in other cases [**?**], so that the use of boundary data is recommended. Even in the outflow condition (8.6), we have found it desirable to maintain a weak relaxation to climatology. The time step is usually chosen so as to be larger than typical turbulent scales (of order 1000 days).

The radiation condition is applied to the model variables : temperature, salinity, tangential and normal velocities. For normal and tangential velocities, u and v, radiation is

applied with phase velocities calculated from u and v respectively. For the radiation of tracers, we use the phase velocity calculated from the tangential velocity in order to avoid calculating too many independent radiation velocities and because tangential velocities and tracers have the same position along the boundary on a C-grid.

8.4.4 Domain decomposition (key_mpp_mpi)

When **key_mpp_mpi** is active in the code, the computational domain is divided into rectangles that are attributed each to a different processor. The open boundary code is "mpp-compatible" up to a certain point. The radiation algorithm will not work if there is an mpp subdomain boundary parallel to the open boundary at the index of the boundary, or the grid point after (outside), or three grid points before (inside). On the other hand, there is no problem if an mpp subdomain boundary cuts the open boundary perpendicularly. These geometrical limitations must be checked for by the user (there is no safeguard in the code). The general principle for the open boundary mpp code is that loops over the open boundaries not sure what this means are performed on local indices (nie0, nie1, nje0, nje1 for an eastern boundary for instance) that are initialized in module *obc_ini.F90*. Those indices have relevant values on the processors that contain a segment of an open boundary. For processors that do not include an open boundary segment, the indices are such that the calculations within the loops are not performed.

Arrays of climatological data that are read from files are seen by all processors and have the same dimensions for all (for instance, for the eastern boundary, uedta(jpjglo,jpk,2)). On the other hand, the arrays for the calculation of radiation are local to each processor (uebnd(jpj,jpk,3,3) for instance). This allowed the CLIPPER model for example, to save on memory where the eastern boundary crossed 8 processors so that *jpj* was much smaller than (*jpjef*-*jpjed*+1).

8.4.5 Volume conservation

It is necessary to control the volume inside a domain when using open boundaries. With fixed boundaries, it is enough to ensure that the total inflow/outflow has reasonable values (either zero or a value compatible with an observed volume balance). When using radiative boundary conditions it is necessary to have a volume constraint because each open boundary works independently from the others. The methodology used to control this volume is identical to the one coded in the ROMS model [**?**].

Explain obc_vol...

OBC algorithm for update, OBC restart, list of routines where obc key appears...

OBC rigid lid ? ...

8.5 Unstructured Open Boundary Conditions (key_bdy) (BDY)

```
!------------------------------------------------------------------------
&nambdy        !  unstructured open boundaries                          ("key_bdy")
!------------------------------------------------------------------------
   cn_mask     =  ''                    !  name of mask file (ln_mask=T)
   cn_dta_frs_T= 'bdydata_grid_T.nc'     !  name of data file (T-points)
   cn_dta_frs_U= 'bdydata_grid_U.nc'     !  name of data file (U-points)
   cn_dta_frs_V= 'bdydata_grid_V.nc'     !  name of data file (V-points)
   cn_dta_fla_T= 'bdydata_bt_grid_T.nc'  !  name of data file for Flather condition (T-points)
   cn_dta_fla_U= 'bdydata_bt_grid_U.nc'  !  name of data file for Flather condition (U-points)
   cn_dta_fla_V= 'bdydata_bt_grid_V.nc'  !  name of data file for Flather condition (V-points)

   ln_clim     = .false.   !  contain 1 (T) or 12 (F) time dumps and be cyclic
   ln_vol      = .false.   !  total volume correction (see volbdy parameter)
   ln_mask     = .false.   !  boundary mask from filbdy_mask (T), boundaries are on edges of domain (F)
   ln_tides    = .false.   !  Apply tidal harmonic forcing with Flather condition
   ln_dyn_fla  = .false.   !  Apply Flather condition to velocities
   ln_tra_frs  = .false.   !  Apply FRS condition to temperature and salinity
   ln_dyn_frs  = .false.   !  Apply FRS condition to velocities
   nn_rimwidth =  9        !  width of the relaxation zone
   nn_dtactl   =  1        !  = 0, bdy data are equal to the initial state
                           !  = 1, bdy data are read in 'bdydata   .nc' files
   nn_volctl   =  0        !  = 0, the total water flux across open boundaries is zero
                           !  = 1, the total volume of the system is conserved
/
```

The BDY module is an alternative implementation of open boundary conditions for regional configurations. It implements the Flow Relaxation Scheme algorithm for temperature, salinity, velocities and ice fields, and the Flather radiation condition for the depth-mean transports. The specification of the location of the open boundary is completely flexible and allows for example the open boundary to follow an isobath or other irregular contour.

The BDY module was modelled on the OBC module and shares many features and a similar coding structure [**?**].

8.5.1 The Flow Relaxation Scheme

The Flow Relaxation Scheme (FRS) [**??**], applies a simple relaxation of the model fields to externally-specified values over a zone next to the edge of the model domain. Given a model prognostic variable Φ

$$\Phi(d) = \alpha(d)\Phi_e(d) + (1 - \alpha(d))\Phi_m(d) \qquad d = 1, N \tag{8.8}$$

where Φ_m is the model solution and Φ_e is the specified external field, d gives the discrete distance from the model boundary and α is a parameter that varies from 1 at $d = 1$ to a small value at $d = N$. It can be shown that this scheme is equivalent to adding a relaxation term to the prognostic equation for Φ of the form :

$$-\frac{1}{\tau}(\Phi - \Phi_e) \tag{8.9}$$

where the relaxation time scale τ is given by a function of α and the model time step Δt :

$$\tau = \frac{1 - \alpha}{\alpha}\Delta t \tag{8.10}$$

Thus the model solution is completely prescribed by the external conditions at the edge of the model domain and is relaxed towards the external conditions over the rest of the FRS zone. The application of a relaxation zone helps to prevent spurious reflection of outgoing signals from the model boundary.

The function α is specified as a $tanh$ function :

$$\alpha(d) = 1 - \tanh\left(\frac{d-1}{2}\right), \quad d = 1, N \tag{8.11}$$

The width of the FRS zone is specified in the namelist as *nn_rimwidth*. This is typically set to a value between 8 and 10.

8.5.2 The Flather radiation scheme

The **?** scheme is a radiation condition on the normal, depth-mean transport across the open boundary. It takes the form

$$U = U_e + \frac{c}{h}(\eta - \eta_e), \tag{8.12}$$

where U is the depth-mean velocity normal to the boundary and η is the sea surface height, both from the model. The subscript e indicates the same fields from external sources. The speed of external gravity waves is given by $c = \sqrt{gh}$, and h is the depth of the water column. The depth-mean normal velocity along the edge of the model domain is set equal to the external depth-mean normal velocity, plus a correction term that allows gravity waves generated internally to exit the model boundary. Note that the sea-surface height gradient in (8.12) is a spatial gradient across the model boundary, so that η_e is defined on the T points with $nbrdta = 1$ and η is defined on the T points with $nbrdta = 2$. U and U_e are defined on the U or V points with $nbrdta = 1$, $i.e.$ between the two T grid points.

8.5.3 Choice of schemes

The Flow Relaxation Scheme may be applied separately to the temperature and salinity (*ln_tra_frs* = true) and the velocity fields (*ln_dyn_frs* = true). Flather radiation conditions may be applied using externally defined barotropic velocities and sea-surface height (*ln_dyn_fla* = true) or using tidal harmonics fields (*ln_tides* = true) or both. FRS and Flather conditions may be applied simultaneously. A typical configuration where all possible conditions might be used is a tidal, shelf-seas model, where the barotropic boundary conditions are fixed with the Flather scheme using tidal harmonics and possibly output from a large-scale model, and FRS conditions are applied to the tracers and baroclinic velocity fields, using fields from a large-scale model.

Note that FRS conditions will work with the filtered (**key_dynspg_flt**) or time-split (**key_dynspg_ts**) solutions for the surface pressure gradient. The Flather condition will only work for the time-split solution (**key_dynspg_ts**). FRS conditions are applied at the end of the main model time step. Flather conditions are applied during the barotropic subcycle in the time-split solution.

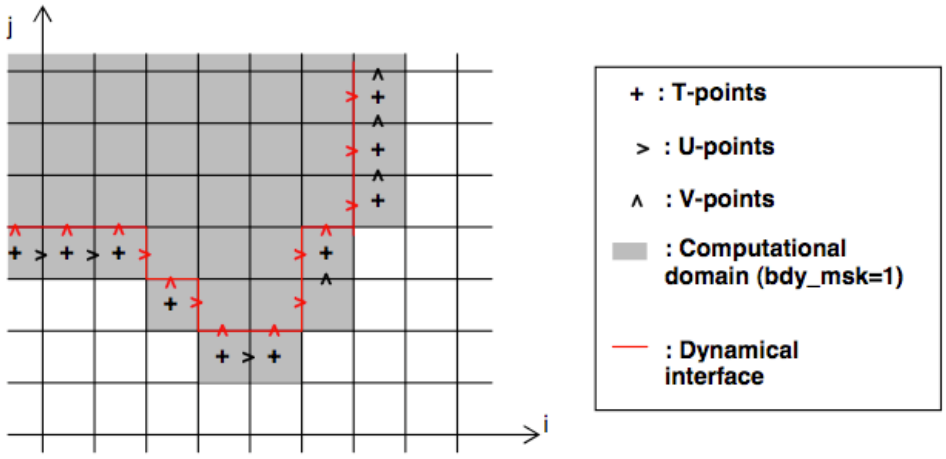

FIG. 8.8 – Example of geometry of unstructured open boundary

8.5.4 Boundary geometry

The definition of the open boundary is completely flexible. An example is shown in Fig. 8.8. The boundary zone is defined by a series of index arrays read in from the input boundary data files : $nbidta$, $nbjdta$, and $nbrdta$. The first two of these define the global (i, j) indices of each point in the boundary zone and the $nbrdta$ array defines the discrete distance from the boundary with $nbrdta = 1$ meaning that the point is next to the edge of the model domain and $nbrdta > 1$ showing that the point is increasingly further away from the edge of the model domain. These arrays are defined separately for each of the T, U and V grids, but the relationship between the points is assumed to be as in Fig. 8.8 with the T points forming the outermost row of the boundary and the first row of velocities normal to the boundary being inside the first row of T points. The order in which the points are defined is unimportant.

8.5.5 Input boundary data files

The input data files for the FRS conditions are defined in the namelist as *cn_dta_frs_T*, *cn_dta_frs_U*, *cn_dta_frs_V*. The input data files for the Flather conditions are defined in the namelist as *cn_dta_fla_T*, *cn_dta_fla_U*, *cn_dta_fla_V*.

The netcdf header of a typical input data file is shown in Fig. 8.9. The file contains the index arrays which define the boundary geometry as noted above and the data arrays for each field. The data arrays are dimensioned on : a time dimension ; xb which is the index of the boundary data point in the horizontal ; and yb which is a degenerate dimension of 1 to enable the file to be read by the standard NEMO I/O routines. The 3D fields also have a depth dimension.

If *ln_clim* is set to *false*, the model expects the units of the time axis to have the form

shown in Fig. 8.9, *i.e.* *"seconds since yyyy-mm-dd hh :mm :ss"* The fields are then linearly interpolated to the model time at each timestep. Note that for this option, the time axis of the input files must completely span the time period of the model integration. If *ln_clim* is set to *true* (climatological boundary forcing), the model will expect either a single set of annual mean fields (constant boundary forcing) or 12 sets of monthly mean fields in the input files.

As in the OBC module there is an option to use initial conditions as boundary conditions. This is chosen by setting *nn_dtactl* = 0. However, since the model defines the boundary geometry by reading the boundary index arrays from the input files, it is still necessary to provide a set of input files in this case. They need only contain the boundary index arrays, $nbidta$, *nbjdta*, *nbrdta*.

8.5.6 Volume correction

There is an option to force the total volume in the regional model to be constant, similar to the option in the OBC module. This is controlled by the *nn_volctl* parameter in the namelist. A value of*nn_volctl* = 0 indicates that this option is not used. If *nn_volctl* = 1 then a correction is applied to the normal velocities around the boundary at each timestep to ensure that the integrated volume flow through the boundary is zero. If *nn_volctl* = 2 then the calculation of the volume change on the timestep includes the change due to the freshwater flux across the surface and the correction velocity corrects for this as well.

8.5.7 Tidal harmonic forcing

To be written....

```
netcdf bdydata_grid_T {
dimensions:
        x = 202 ;
        y = 104 ;
        xb = 2058 ;
        yb = 1 ;
        depth = 40 ;
        time_counter = UNLIMITED ; // (61 currently)
variables:
        float nav_lon(y, x) ;
                nav_lon:units = "degrees_east" ;
                nav_lon:valid_min = 0.f ;
                nav_lon:valid_max = 1005.f ;
                nav_lon:long_name = "Longitude" ;
        float nav_lat(y, x) ;
                nav_lat:units = "degrees_north" ;
                nav_lat:valid_min = 0.f ;
                nav_lat:valid_max = 515.f ;
                nav_lat:long_name = "Latitude" ;
        float time_counter(time_counter) ;
                time_counter:units = "seconds since 1992-01-01 00:00:00" ;
                time_counter:calendar = "noleap" ;
                time_counter:title = "Time" ;
                time_counter:long_name = "Time axis" ;
                time_counter:time_counter_origin = "1992-Jan-01 00:00:00" ;
        float depth(depth) ;
                depth:units = "model_levels" ;
                depth:valid_min = 59.9493f ;
                depth:valid_max = 4042.771f ;
                depth:long_name = "Model levels" ;
        float bdy_msk(y, x) ;
                bdy_msk:units = "unitless" ;
                bdy_msk:missing_value = 1.e+20f ;
                bdy_msk:long_name = "Unstructured boundary mask" ;
        int nbidta(yb, xb) ;
                nbidta:units = "unitless" ;
                nbidta:valid_min = 2 ;
                nbidta:valid_max = 200 ;
                nbidta:long_name = "Bdy i indices" ;
        int nbjdta(yb, xb) ;
                nbjdta:units = "unitless" ;
                nbjdta:valid_min = 3 ;
                nbjdta:valid_max = 102 ;
                nbjdta:long_name = "Bdy j indices" ;
        int nbrdta(yb, xb) ;
                nbrdta:units = "unitless" ;
                nbrdta:valid_min = 1 ;
                nbrdta:valid_max = 10 ;
                nbrdta:long_name = "Bdy discrete distance" ;
        float votemper(time_counter, depth, yb, xb) ;
                votemper:units = "C" ;
                votemper:missing_value = 0.f ;
                votemper:valid_min = 10.f ;
                votemper:valid_max = 10.f ;
                votemper:long_name = "Temperature" ;
                votemper:short_name = "votemper" ;
        float vosaline(time_counter, depth, yb, xb) ;
                vosaline:units = "PSU" ;
                vosaline:missing_value = 0.f ;
                vosaline:valid_min = 35.5f ;
                vosaline:valid_max = 35.5f ;
                vosaline:long_name = "Salinity" ;
                vosaline:short_name = "vosaline" ;

// global attributes:
                :title = "Unstructured boundaries data file at T-points" ;
                :history = "01-Mar-2005 11:31:44" ;
                :institution = "GIP MERCATOR OCEAN" ;
                :references = "http://www.mercator-ocean.fr" ;
                :rimwidth = 10 ;
}
```

FIG. 8.9 – Example of header of netcdf input data file for BDY

9 Lateral Ocean Physics (LDF)

Contents

The lateral physics terms in the momentum and tracer equations have been described in §2.5.1 and their discrete formulation in §5.2 and §6.6). In this section we further discuss each lateral physics option. Choosing one lateral physics scheme means for the user defining, (1) the space and time variations of the eddy coefficients ; (2) the direction along which the lateral diffusive fluxes are evaluated (model level, geopotential or isopycnal surfaces) ; and (3) the type of operator used (harmonic, or biharmonic operators, and for tracers only, eddy induced advection on tracers). These three aspects of the lateral diffusion are set through namelist parameters and CPP keys (see the *nam_traldf* and *nam_dynldf* below).

```
!-----------------------------------------------------------------------
&namtra_ldf    !   lateral diffusion scheme for tracer
!-----------------------------------------------------------------------
   !                     !  Type of the operator :
   ln_traldf_lap    =  .true.   !  laplacian operator
   ln_traldf_bilap  =  .false.  !  bilaplacian operator
   !                     !  Direction of action  :
   ln_traldf_level  =  .false.  !  iso-level
   ln_traldf_hor    =  .false.  !  horizontal (geopotential)           (require "key_ldfslp" when ln_sco=T)
   ln_traldf_iso    =  .true.   !  iso-neutral                        (require "key_ldfslp")
   ln_traldf_grif   =  .false.  !  griffies skew flux formulation     (require "key_ldfslp")  ! UNDER TEST, DO NOT USE
   ln_traldf_gdia   =  .false.  !  griffies operator strfn diagnostics  (require "key_ldfslp")  ! UNDER TEST, DO NOT USE
   !                     !  Coefficient
   rn_aht_0         =  2000.    !  horizontal eddy diffusivity for tracers [m2/s]
   rn_ahtb_0        =     0.    !  background eddy diffusivity for ldf_iso [m2/s]
   rn_aeiv_0        =  2000.    !  eddy induced velocity coefficient [m2/s]    (require "key_traldf_eiv")
/


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

9.1 Lateral Mixing Coefficient (*ldftra.F90*, *ldfdyn.F90*)

Introducing a space variation in the lateral eddy mixing coefficients changes the model core memory requirement, adding up to four extra three-dimensional arrays for the geopotential or isopycnal second order operator applied to momentum. Six CPP keys control the space variation of eddy coefficients : three for momentum and three for tracer. The three choices allow : a space variation in the three space directions (**key_traldf_c3d**, **key_dynldf_c3d**), in the horizontal plane (**key_traldf_c2d**, **key_dynldf_c2d**), or in the vertical only (**key_traldf_c1d**, **key_dynldf_c1d**). The default option is a constant value over the whole ocean on both momentum and tracers.

The number of additional arrays that have to be defined and the gridpoint position at which they are defined depend on both the space variation chosen and the type of operator

used. The resulting eddy viscosity and diffusivity coefficients can be a function of more than one variable. Changes in the computer code when switching from one option to another have been minimized by introducing the eddy coefficients as statement functions (include file *ldftra_substitute.h90* and *ldfdyn_substitute.h90*). The functions are replaced by their actual meaning during the preprocessing step (CPP). The specification of the space variation of the coefficient is made in *ldftra.F90* and *ldfdyn.F90*, or more precisely in include files *traldf_cNd.h90* and *dynldf_cNd.h90*, with N=1, 2 or 3. The user can modify these include files as he/she wishes. The way the mixing coefficient are set in the reference version can be briefly described as follows :

Constant Mixing Coefficients (default option)

When none of the **key_dynldf_...** and **key_traldf_...** keys are defined, a constant value is used over the whole ocean for momentum and tracers, which is specified through the *rn_ahm0* and *rn_aht0* namelist parameters.

Vertically varying Mixing Coefficients (key_traldf_c1d and key_dynldf_c1d)

The 1D option is only available when using the z-coordinate with full step. Indeed in all the other types of vertical coordinate, the depth is a 3D function of (**i**,**j**,**k**) and therefore, introducing depth-dependent mixing coefficients will require 3D arrays. In the 1D option, a hyperbolic variation of the lateral mixing coefficient is introduced in which the surface value is *rn_aht0* (*rn_ahm0*), the bottom value is 1/4 of the surface value, and the transition takes place around z=300 m with a width of 300 m ($i.e.$ both the depth and the width of the inflection point are set to 300 m). This profile is hard coded in file *traldf_c1d.h90*, but can be easily modified by users.

Horizontally Varying Mixing Coefficients (key_traldf_c2d and key_dynldf_c2d)

By default the horizontal variation of the eddy coefficient depends on the local mesh size and the type of operator used :

$$A_l = \begin{cases} \dfrac{\max(e_1, e_2)}{e_{max}} A_o^l & \text{for laplacian operator} \\ \dfrac{\max(e_1, e_2)^3}{e_{max}^3} A_o^l & \text{for bilaplacian operator} \end{cases} \tag{9.1}$$

where e_{max} is the maximum of e_1 and e_2 taken over the whole masked ocean domain, and A_o^l is the *rn_ahm0* (momentum) or *rn_aht0* (tracer) namelist parameter. This variation is intended to reflect the lesser need for subgrid scale eddy mixing where the grid size is smaller in the domain. It was introduced in the context of the DYNAMO modelling project [**?**]. Note that such a grid scale dependance of mixing coefficients significantly increase the range of stability of model configurations presenting large changes in grid pacing such as global ocean models. Indeed, in such a case, a constant mixing coefficient

can lead to a blow up of the model due to large coefficient compare to the smallest grid size (see §3.3), especially when using a bilaplacian operator.

Other formulations can be introduced by the user for a given configuration. For example, in the ORCA2 global ocean model (**key_orca_r2**), the laplacian viscosity operator uses *rn_ahm0* = 4.10^4 m^2/s poleward of 20° north and south and decreases linearly to *rn_aht0* = 2.10^3 m^2/s at the equator [**??**]. This modification can be found in routine *ldf_dyn_c2d_orca* defined in *ldfdyn_c2d.F90*. Similar modified horizontal variations can be found with the Antarctic or Arctic sub-domain options of ORCA2 and ORCA05 (**key_antarctic** or **key_arctic** defined, see *ldfdyn_antarctic.h90* and *ldfdyn_arctic.h90*).

Space Varying Mixing Coefficients (key_traldf_c3d and key_dynldf_c3d)

The 3D space variation of the mixing coefficient is simply the combination of the 1D and 2D cases, $i.e.$ a hyperbolic tangent variation with depth associated with a grid size dependence of the magnitude of the coefficient.

Space and Time Varying Mixing Coefficients

There is no default specification of space and time varying mixing coefficient. The only case available is specific to the ORCA2 and ORCA05 global ocean configurations (**key_orca_r2** or **key_orca_r05**). It provides only a tracer mixing coefficient for eddy induced velocity (ORCA2) or both iso-neutral and eddy induced velocity (ORCA05) that depends on the local growth rate of baroclinic instability. This specification is actually used when an ORCA key and both **key_traldf_eiv** and **key_traldf_c2d** are defined.

A space variation in the eddy coefficient appeals several remarks :

(1) the momentum diffusion operator acting along model level surfaces is written in terms of curl and divergent components of the horizontal current (see §2.5.2). Although the eddy coefficient can be set to different values in these two terms, this option is not available.

(2) with an horizontally varying viscosity, the quadratic integral constraints on enstrophy and on the square of the horizontal divergence for operators acting along model-surfaces are no longer satisfied (Appendix C.7).

(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 *rn_ahmb0* or *rn_ahtb0*, a background horizontal eddy viscosity or diffusivity coefficient (namelist parameters whose default values are 0). However, the technique used to compute the isopycnal slopes is intended to get rid of such a background diffusion, since it introduces spurious diapycnal diffusion (see §9.2).

(4) when an eddy induced advection term is used (**key_traldf_eiv**), A^{eiv}, the eddy induced coefficient has to be defined. Its space variations are controlled by the same CPP variable as for the eddy diffusivity coefficient ($i.e.$ **key_traldf_cNd**).

(5) the eddy coefficient associated with a biharmonic operator must be set to a *negative* value.

(6) it is possible to use both the laplacian and biharmonic operators concurrently.

(7) it is possible to run without explicit lateral diffusion on momentum (*ln_dynldf_lap* = *ln_dynldf_bilap* = false). This is recommended when using the UBS advection scheme on momentum (*ln_dynadv_ubs* = true, see 6.3.2) and can be useful for testing purposes.

9.2 Direction of Lateral Mixing (*ldfslp.F90*)

A direction for lateral mixing has to be defined when the desired operator does not act along the model levels. This occurs when (a) horizontal mixing is required on tracer or momentum (*ln_traldf_hor* or *ln_dynldf_hor*) in s- or mixed s-z- coordinates, and (b) isoneutral mixing is required whatever the vertical coordinate is. This direction of mixing is defined by its slopes in the **i**- and **j**-directions at the face of the cell of the quantity to be diffused. For a tracer, this leads to the following four slopes : r_{1u}, r_{1w}, r_{2v}, r_{2w} (see (5.10)), 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.

9.2.1 slopes for tracer geopotential mixing in the s-coordinate

In s-coordinates, geopotential mixing ($i.e.$ horizontal mixing) r_1 and r_2 are the slopes between the geopotential and computational surfaces. Their discrete formulation is found by locally solving (5.10) when the diffusive fluxes in the three directions are set to zero and T is assumed to be horizontally uniform, $i.e.$ a linear function of z_T, the depth of a T-point.

$$
\begin{aligned}
r_{1u} &= \frac{e_{3u}}{\left(e_{1u}\,\overline{\overline{e_{3w}}}^{\,i+1/2,\,k}\right)}\,\delta_{i+1/2}[z_t] &&\approx \frac{1}{e_{1u}}\,\delta_{i+1/2}[z_t] \\
r_{2v} &= \frac{e_{3v}}{\left(e_{2v}\,\overline{\overline{e_{3w}}}^{\,j+1/2,\,k}\right)}\,\delta_{j+1/2}[z_t] &&\approx \frac{1}{e_{2v}}\,\delta_{j+1/2}[z_t] \\
r_{1w} &= \frac{1}{e_{1w}}\,\overline{\overline{\delta_{i+1/2}[z_t]}}^{\,i,\,k+1/2} &&\approx \frac{1}{e_{1w}}\,\delta_{i+1/2}[z_{uw}] \\
r_{2w} &= \frac{1}{e_{2w}}\,\overline{\overline{\delta_{j+1/2}[z_t]}}^{\,j,\,k+1/2} &&\approx \frac{1}{e_{2w}}\,\delta_{j+1/2}[z_{vw}]
\end{aligned}
\tag{9.2}
$$

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

9.2.2 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 (5.10) and setting the diffusive fluxes in the three directions to zero leads to the following definition for the neutral slopes :

$$
\begin{aligned}
r_{1u} &= \frac{e_{3u}}{e_{1u}} \frac{\delta_{i+1/2}[\rho]}{\overline{\overline{\delta_{k+1/2}[\rho]}}^{i+1/2,\,k}} \\
r_{2v} &= \frac{e_{3v}}{e_{2v}} \frac{\delta_{j+1/2}[\rho]}{\overline{\overline{\delta_{k+1/2}[\rho]}}^{j+1/2,\,k}} \\
r_{1w} &= \frac{e_{3w}}{e_{1w}} \frac{\overline{\overline{\delta_{i+1/2}[\rho]}}^{i,\,k+1/2}}{\delta_{k+1/2}[\rho]} \\
r_{2w} &= \frac{e_{3w}}{e_{2w}} \frac{\overline{\overline{\delta_{j+1/2}[\rho]}}^{j,\,k+1/2}}{\delta_{k+1/2}[\rho]}
\end{aligned}
\tag{9.3}
$$

As the mixing is performed along neutral surfaces, the gradient of ρ in (9.3) has to be evaluated at the same local pressure (which, in decibars, is approximated by the depth in meters in the model). Therefore (9.3) cannot be used as such, but further transformation is needed depending on the vertical coordinate used :

z-coordinate with full step : in (9.3) the densities appearing in the i and j derivatives are taken at the same depth, thus the $insitu$ density can be used. This is not the case for the vertical derivatives : $\delta_{k+1/2}[\rho]$ is replaced by $-\rho N^2/g$, where N^2 is the local Brunt-Vaisälä frequency evaluated following **?** (see §5.8.2).

z-coordinate with partial step : this case is identical to the full step case except that at partial step level, the *horizontal* density gradient is evaluated as described in §5.9.

s- or hybrid s-z- coordinate : in the current release of *NEMO* , there is no specific treatment for iso-neutral mixing in the s-coordinate. In other words, iso-neutral mixing will only be accurately represented with a linear equation of state (*nn_eos*=1 or 2). In the case of a "true" equation of state, the evaluation of i and j derivatives in (9.3) will include a pressure dependent part, leading to the wrong evaluation of the neutral slopes.

Note : The solution for s-coordinate passes trough the use of different (and better) expression for the constraint on iso-neutral fluxes. Following **?**, instead of specifying directly that there is a zero neutral diffusive flux of locally referenced potential density, we stay in the T-S plane and consider the balance between the neutral direction diffusive fluxes of potential temperature and salinity :

$$
\alpha \, \mathbf{F}(T) = \beta \, \mathbf{F}(S) \tag{9.4}
$$

This constraint leads to the following definition for the slopes :

$$
\begin{aligned}
r_{1u} &= \frac{e_{3u}}{e_{1u}} \frac{\alpha_u \, \delta_{i+1/2}[T] - \beta_u \, \delta_{i+1/2}[S]}{\alpha_u \, \overline{\overline{\delta_{k+1/2}[T]}}^{\,i+1/2,\,k} - \beta_u \, \overline{\overline{\delta_{k+1/2}[S]}}^{\,i+1/2,\,k}} \\
r_{2v} &= \frac{e_{3v}}{e_{2v}} \frac{\alpha_v \, \delta_{j+1/2}[T] - \beta_v \, \delta_{j+1/2}[S]}{\alpha_v \, \overline{\overline{\delta_{k+1/2}[T]}}^{\,j+1/2,\,k} - \beta_v \, \overline{\overline{\delta_{k+1/2}[S]}}^{\,j+1/2,\,k}} \\
r_{1w} &= \frac{e_{3w}}{e_{1w}} \frac{\alpha_w \, \overline{\overline{\delta_{i+1/2}[T]}}^{\,i,\,k+1/2} - \beta_w \, \overline{\overline{\delta_{i+1/2}[S]}}^{\,i,\,k+1/2}}{\alpha_w \, \delta_{k+1/2}[T] - \beta_w \, \delta_{k+1/2}[S]} \\
r_{2w} &= \frac{e_{3w}}{e_{2w}} \frac{\alpha_w \, \overline{\overline{\delta_{j+1/2}[T]}}^{\,j,\,k+1/2} - \beta_w \, \overline{\overline{\delta_{j+1/2}[S]}}^{\,j,\,k+1/2}}{\alpha_w \, \delta_{k+1/2}[T] - \beta_w \, \delta_{k+1/2}[S]}
\end{aligned}
\tag{9.5}
$$

where α and β, the thermal expansion and saline contraction coefficients introduced in §5.8.2, have to be evaluated at the three velocity points. In order to save computation time, they should be approximated by the mean of their values at T-points (for example in the case of α : $\alpha_u = \overline{\alpha_T}^{\,i+1/2}$, $\alpha_v = \overline{\alpha_T}^{\,j+1/2}$ and $\alpha_w = \overline{\alpha_T}^{\,k+1/2}$). Note that such a formulation could be also used in the z-coordinate and z-coordinate with partial steps cases.

This implementation is a rather old one. It is similar to the one proposed by Cox [1987], except for the background horizontal diffusion. Indeed, the Cox implementation of isopycnal diffusion in GFDL-type models requires a minimum background horizontal diffusion for numerical stability reasons. To overcome this problem, several techniques have been proposed in which the numerical schemes of the ocean model are modified [**??**]. Here, another strategy has been chosen [**?**] : a local filtering of the iso-neutral slopes (made on 9 grid-points) prevents the development of grid point noise generated by the iso-neutral diffusion operator (Fig. 9.1). This allows an iso-neutral diffusion scheme without additional background horizontal mixing. This technique can be viewed as a diffusion operator that acts along large-scale (2 Δx) iso-neutral surfaces. The diapycnal diffusion required for numerical stability is thus minimized and its net effect on the flow is quite small when compared to the effect of an horizontal background mixing.

Nevertheless, this iso-neutral operator does not ensure that variance cannot increase, contrary to the **?** operator which has that property.

In addition and also for numerical stability reasons [**??**], the slopes are bounded by $1/100$ everywhere. This limit is decreasing linearly to zero fom 70 meters depth and the surface (the fact that the eddies "feel" the surface motivates this flattening of isopycnals near the surface).

For numerical stability reasons [**??**], the slopes must also be bounded by $1/100$ everywhere. This constraint is applied in a piecewise linear fashion, increasing from zero at the surface to $1/100$ at 70 metres and thereafter decreasing to zero at the bottom of the ocean. (the fact that the eddies "feel" the surface motivates this flattening of isopycnals near the surface).

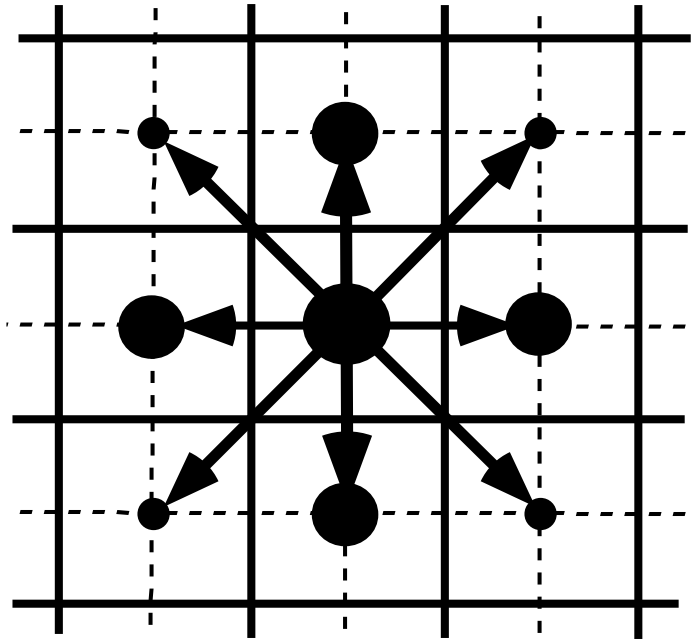

FIG. 9.1 – averaging procedure for isopycnal slope computation.

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

9.2.3 slopes for momentum iso-neutral mixing

The iso-neutral diffusion operator on momentum is the same as the one used on tracers but applied to each component of the velocity separately (see (6.26) in section 6.6.2). The slopes between the surface along which the diffusion operator acts and the surface of computation (z- or s-surfaces) are defined at T-, f-, and uw- points for the u-component, and T-, f- and vw- points for the v-component. They are computed from the slopes used for tracer diffusion, *i.e.* (9.2) and (9.3) :

$$\begin{aligned} r_{1t} &= \overline{r_{1u}}^{\,i} & & & r_{1f} &= \overline{r_{1u}}^{\,i+1/2} \\ r_{2f} &= \overline{r_{2v}}^{\,j+1/2} & & & r_{2t} &= \overline{r_{2v}}^{\,j} \\ r_{1uw} &= \overline{r_{1w}}^{\,i+1/2} & \text{and} & & r_{1vw} &= \overline{r_{1w}}^{\,j+1/2} \\ r_{2uw} &= \overline{r_{2w}}^{\,j+1/2} & & & r_{2vw} &= \overline{r_{2w}}^{\,j+1/2} \end{aligned} \tag{9.6}$$

The major issue remaining is in the specification of the boundary conditions. The same boundary conditions are chosen as those used for lateral diffusion along model level surfaces, i.e. using the shear computed along the model levels and with no additional friction at the ocean bottom (see §8.1).

9.3 Eddy Induced Velocity (*traadv_eiv.F90*, *ldfeiv.F90*)

When Gent and McWilliams [1990] diffusion is used (**key_traldf_eiv** defined), an eddy induced tracer advection term is added, the formulation of which depends on the slopes of iso-neutral surfaces. Contrary to the case of iso-neutral mixing, the slopes used here are referenced to the geopotential surfaces, $i.e.$ (9.2) is used in z-coordinates, and the sum (9.2) + (9.3) in s-coordinates. The eddy induced velocity is given by :

$$
\begin{aligned}
u^* &= \frac{1}{e_{2u}e_{3u}}\,\delta_k\left[e_{2u}\,A^{eiv}_{uw}\,\overline{r_{1w}}^{\,i+1/2}\right] \\
v^* &= \frac{1}{e_{1u}e_{3v}}\,\delta_k\left[e_{1v}\,A^{eiv}_{vw}\,\overline{r_{2w}}^{\,j+1/2}\right] \\
w^* &= \frac{1}{e_{1w}e_{2w}}\,\left\{\delta_i\left[e_{2u}\,A^{eiv}_{uw}\,\overline{r_{1w}}^{\,i+1/2}\right]+\delta_j\left[e_{1v}\,A^{eiv}_{vw}\,\overline{r_{2w}}^{\,j+1/2}\right]\right\}
\end{aligned}
\tag{9.7}
$$

where A^{eiv} is the eddy induced velocity coefficient whose value is set through *rn_aeiv*, a *nam_traldf* namelist parameter. The three components of the eddy induced velocity are computed and add to the eulerian velocity in *traadv_eiv.F90*. This has been preferred to a separate computation of the advective trends associated with the eiv velocity, since it allows us to take advantage of all the advection schemes offered for the tracers (see §5.1) and not just the 2^{nd} order advection scheme as in previous releases of OPA [**?**]. This is particularly useful for passive tracers where *positivity* of the advection scheme is of paramount importance.

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

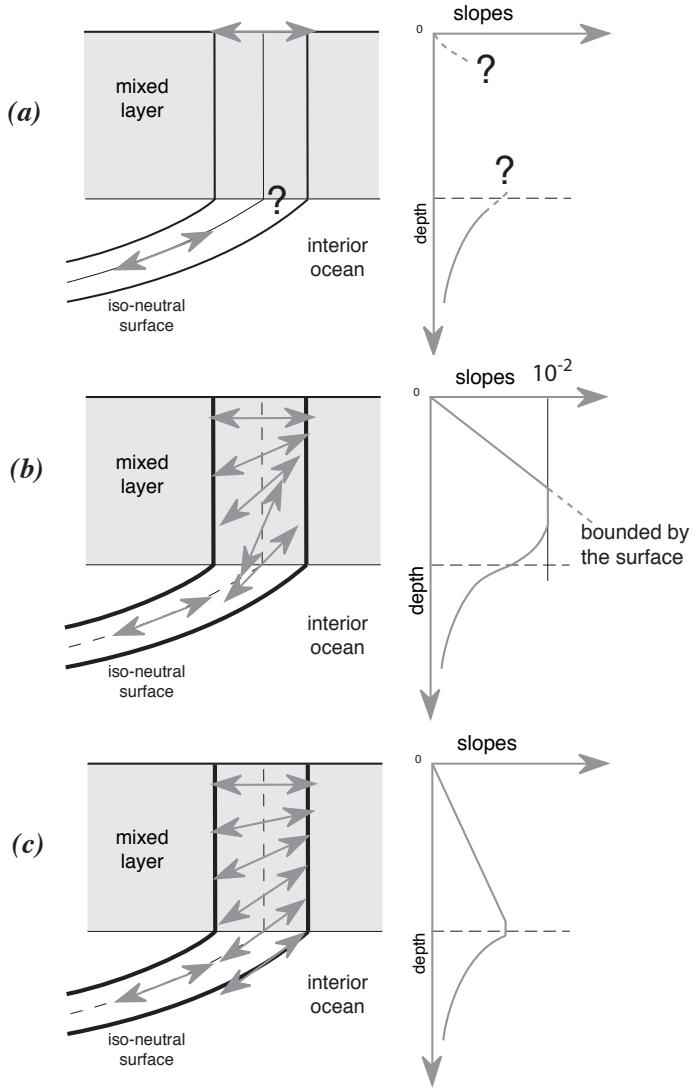

FIG. 9.2 – Vertical profile of the slope used for lateral mixing in the mixed layer : *(a)* in the real ocean the slope is the iso-neutral slope in the ocean interior, which has to be adjusted at the surface boundary (i.e. it must tend to zero at the surface since there is no mixing across the air-sea interface : wall boundary condition). Nevertheless, the profile between the surface zero value and the interior iso-neutral one is unknown, and especially the value at the base of the mixed layer ; *(b)* profile of slope using a linear tapering of the slope near the surface and imposing a maximum slope of 1/100 ; *(c)* profile of slope actually used in *NEMO* : a linear decrease of the slope from zero at the surface to its ocean interior value computed just below the mixed layer. Note the huge change in the slope at the base of the mixed layer between *(b)* and *(c)*.

10 Vertical Ocean Physics (ZDF)

Contents

10.1 Vertical Mixing

The discrete form of the ocean subgrid scale physics has been presented in §5.3 and §6.7. 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. 7), 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 §5.4.3), and specified through a bottom friction parameterisation for momentum (see §10.4).

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 §5.3 and §6.7). These coefficients can be assumed to be either constant, or a function of the local Richardson number, or computed from a turbulent closure model (either TKE or KPP formulation). The computation of these coefficients is initialized in the *zdfini.F90* module and performed in the *zdfric.F90*, *zdftke.F90* or *zdfkpp.F90* modules. The trends due to the vertical momentum and tracer diffusion, including the surface forcing, are computed and added to the general trend in the *dynzdf.F90* and *trazdf.F90* modules, respectively. These trends can be computed using either a forward time stepping scheme (namelist parameter *ln_zdfexp*=true) or a backward time stepping scheme (*ln_zdfexp*=false) depending on the magnitude of the mixing coefficients, and thus of the formulation used (see §3).

10.1.1 Constant (key_zdfcst)

```
!-----------------------------------------------------------------------
&namzdf        !   vertical physics
!-----------------------------------------------------------------------
   rn_avm0     =   1.2e-4  !  vertical eddy viscosity   [m2/s]          (background Kz if not "key_zdfcst")
   rn_avt0     =   1.2e-5  !  vertical eddy diffusivity [m2/s]          (background Kz if not "key_zdfcst")
   nn_avb      =    0      !  profile for background avt & avm (=1) or not (=0)
   nn_havtb    =    0      !  horizontal shape for avtb (=1) or not (=0)
   ln_zdfevd   = .true.    !  enhanced vertical diffusion (evd) (T) or not (F)
   nn_evdm     =    0      !  evd apply on tracer (=0) or on tracer and momentum (=1)
   rn_avevd    =  100.     !  evd mixing coefficient [m2/s]
   ln_zdfnpc   = .false.   !  Non-Penetrative Convective algorithm (T) or not (F)
   nn_npc      =    1           !  frequency of application of npc
   nn_npcp     =  365           !  npc control print frequency
   ln_zdfexp   = .false.   !  time-stepping: split-explicit (T) or implicit (F) time stepping
   nn_zdfexp   =    3           !  number of sub-timestep for ln_zdfexp=T
/
```

When **key_zdfcst** is defined, the momentum and tracer vertical eddy coefficients are set to constant values over the whole ocean. This is the crudest way to define the vertical ocean physics. It is recommended that this option is only used in process studies, not in basin scale simulations. Typical values used in this case are :

$$A_u^{vm} = A_v^{vm} = 1.2\ 10^{-4}\ m^2.s^{-1}$$
$$A^{vT} = A^{vS} = 1.2\ 10^{-5}\ m^2.s^{-1}$$

These values are set through the *rn_avm0* and *rn_avt0* namelist parameters. In all cases, do not use values smaller that those associated with the molecular viscosity and diffusivity, that is $\sim 10^{-6}\ m^2.s^{-1}$ for momentum, $\sim 10^{-7}\ m^2.s^{-1}$ for temperature and $\sim 10^{-9}\ m^2.s^{-1}$ for salinity.

10.1.2 Richardson Number Dependent (key_zdfric)

```
!-----------------------------------------------------------------------
&namzdf_ric    !   richardson number dependent vertical diffusion       ("key_zdfric" )
!-----------------------------------------------------------------------
   rn_avmri    = 100.e-4   !  maximum value of the vertical viscosity
   rn_alp      =   5.      !  coefficient of the parameterization
   nn_ric      =   2       !  coefficient of the parameterization
/
```

When **key_zdfric** is defined, a local Richardson number dependent formulation for the vertical momentum and tracer eddy coefficients is set. The vertical mixing coefficients are diagnosed from the large scale variables computed by the model. *In situ* measurements have been used to link vertical turbulent activity to large scale ocean structures. The hypothesis of a mixing mainly maintained by the growth of Kelvin-Helmholtz like instabilities leads to a dependency between the vertical eddy coefficients and the local Richardson number ($i.e.$ the ratio of stratification to vertical shear). Following **?**, the following formulation has been implemented :

$$\begin{cases} A^{vT} = \dfrac{A^{vT}_{ric}}{(1 + a\ Ri)^n} + A^{vT}_b \\ A^{vm} = \dfrac{A^{vT}}{(1 + a\ Ri)} + A^{vm}_b \end{cases} \tag{10.1}$$

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 §5.8.2), A^{vT}_b and A^{vm}_b are the constant background values set as in the constant case (see §10.1.1), and $A^{vT}_{ric} = 10^{-4}\ m^2.s^{-1}$ is the maximum value that can be reached by the coefficient when $Ri \leq 0$, $a = 5$ and $n = 2$. The last three values can be modified by setting the *rn_avmri*, *rn_alp* and *nn_ric* namelist parameters, respectively.

10.1.3 TKE Turbulent Closure Scheme (key_zdftke)

```
!-----------------------------------------------------------------------
&namzdf_tke    !   turbulent eddy kinetic dependent vertical diffusion  ("key_zdftke")
!-----------------------------------------------------------------------
   rn_ediff    =   0.1     !  coef. for vertical eddy coef. (avt=rn_ediff*mxl*sqrt(e) )
   rn_ediss    =   0.7     !  coef. of the Kolmogoroff dissipation
   rn_ebb      =  67.83    !  coef. of the surface input of tke (=67.83 suggested when ln_mxl0=T)
   rn_emin     =   1.e-6   !  minimum value of tke [m2/s2]
   rn_emin0    =   1.e-4   !  surface minimum value of tke [m2/s2]
   nn_mxl      =   2       !  mixing length: = 0 bounded by the distance to surface and bottom
                           !                 = 1 bounded by the local vertical scale factor
                           !                 = 2 first vertical derivative of mixing length bounded by 1
                           !                 = 3 as =2 with distinct disspipative an mixing length scale
   nn_pdl      =   1       !  Prandtl number function of richarson number (=1, avt=pdl(Ri)*avm) or not (=0, avt=avm)
   ln_mxl0     = .true.    !  surface mixing length scale = F(wind stress) (T) or not (F)
   rn_mxl0     =   0.04    !  surface  buoyancy lenght scale minimum value
   ln_lc       = .true.    !  Langmuir cell parameterisation (Axell 2002)
   rn_lc       =   0.15    !  coef. associated to Langmuir cells
   nn_etau     =   1       !  penetration of tke below the mixed layer (ML) due to internal & intertial waves
                           !        = 0 no penetration
                           !        = 1 add a tke source below the ML
                           !        = 2 add a tke source just at the base of the ML
```

```
                              !       = 3 as = 1 applied on HF part of the stress    ("key_coupled")
   rn_efr      =   0.05       !  fraction of surface tke value which penetrates below the ML (nn_etau=1 or 2)
   nn_htau     =   1          !  type of exponential decrease of tke penetration below the ML
                              !       = 0  constant 10 m length scale
                              !       = 1  0.5m at the equator to 30m poleward of 40 degrees
/
```

The vertical eddy viscosity and diffusivity coefficients are computed from a TKE turbulent closure model based on a prognostic equation for $\bar{e}$, the turbulent kinetic energy, and a closure assumption for the turbulent length scales. This turbulent closure model has been developed by **?** in the atmospheric case, adapted by **?** for the oceanic case, and embedded in OPA, the ancestor of NEMO, by **?** for equatorial Atlantic simulations. Since then, significant modifications have been introduced by **?** in both the implementation and the formulation of the mixing length scale. The time evolution of $\bar{e}$ is the result of the production of $\bar{e}$ through vertical shear, its destruction through stratification, its vertical diffusion, and its dissipation of **?** type :

$$\frac{\partial \bar{e}}{\partial t} = \frac{K_m}{{e_3}^2} \left[\left(\frac{\partial u}{\partial k} \right)^2 + \left(\frac{\partial v}{\partial k} \right)^2 \right] - K_\rho \, N^2 + \frac{1}{e_3} \frac{\partial}{\partial k} \left[\frac{A^{vm}}{e_3} \frac{\partial \bar{e}}{\partial k} \right] - c_\epsilon \frac{\bar{e}^{3/2}}{l_\epsilon} \tag{10.2}$$

$$\begin{aligned} K_m &= C_k \, l_k \sqrt{\bar{e}} \\ K_\rho &= A^{vm} / P_{rt} \end{aligned} \tag{10.3}$$

where N is the local Brunt-Vaisälä frequency (see §5.8.2), l_ϵ and l_κ are the dissipation and mixing length scales, P_{rt} is the Prandtl number, K_m and K_ρ are the vertical eddy viscosity and diffusivity coefficients. The constants $C_k = 0.1$ and $C_\epsilon = \sqrt{2}/2 \approx 0.7$ are designed to deal with vertical mixing at any depth [**?**]. They are set through namelist parameters *nn_ediff* and *nn_ediss*. P_{rt} can be set to unity or, following **?**, be a function of the local Richardson number, R_i :

$$P_{rt} = \begin{cases} 1 & \text{if } R_i \leq 0.2 \\ 5 \, R_i & \text{if } 0.2 \leq R_i \leq 2 \\ 10 & \text{if } 2 \leq R_i \end{cases}$$

The choice of P_{rt} is controlled by the *nn_pdl* namelist parameter.

At the sea surface, the value of $\bar{e}$ is prescribed from the wind stress field as $\bar{e}_o = e_{bb}|\tau|/\rho_o$, with e_{bb} the *rn_ebb* namelist parameter. The default value of e_{bb} is 3.75. [**?**]), however a much larger value can be used when taking into account the surface wave breaking (see below Eq. (10.8)). The bottom value of TKE is assumed to be equal to the value of the level just above. The time integration of the $\bar{e}$ equation may formally lead to negative values because the numerical scheme does not ensure its positivity. To overcome this problem, a cut-off in the minimum value of $\bar{e}$ is used (*rn_emin* namelist parameter). Following **?**, the cut-off value is set to $\sqrt{2}/2 \; 10^{-6} \; m^2.s^{-2}$. This allows the subsequent formulations to match that of **?** for the diffusion in the thermocline and deep ocean : $K_\rho = 10^{-3}/N$. In addition, a cut-off is applied on K_m and K_ρ 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 *rn_avm0* and *rn_avt0* (namzdf namelist, see §10.1.1).

Turbulent length scale

For computational efficiency, the original formulation of the turbulent length scales proposed by **?** has been simplified. Four formulations are proposed, the choice of which is controlled by the *nn_mxl* namelist parameter. The first two are based on the following first order approximation [**?**] :

$$l_k = l_\epsilon = \sqrt{2\bar{e}}\,/N \tag{10.4}$$

which is valid in a stable stratified region with constant values of the Brunt- Vaisälä frequency. The resulting length scale is bounded by the distance to the surface or to the bottom (*nn_mxl* = 0) or by the local vertical scale factor (*nn_mxl* = 1). **?** notice that this simplification has two major drawbacks : it makes no sense for locally unstable stratification and the computation no longer uses all the information contained in the vertical density profile. To overcome these drawbacks, **?** introduces the *nn_mxl* = 2 or 3 cases, which add an extra assumption concerning the vertical gradient of the computed length scale. So, the length scales are first evaluated as in (10.4) and then bounded such that :

$$\frac{1}{e_3}\left|\frac{\partial l}{\partial k}\right| \leq 1 \qquad \text{with } l = l_k = l_\epsilon \tag{10.5}$$

(10.5) means that the vertical variations of the length scale cannot be larger than the variations of depth. It provides a better approximation of the **?** formulation while being much less time consuming. In particular, it allows the length scale to be limited not only by 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. 10.1). In order to impose the (10.5) constraint, we introduce two additional length scales : l_{up} and l_{dwn}, the upward and downward length scales, and evaluate the dissipation and mixing length scales as (and note that here we use numerical indexing) :

$$\begin{aligned} l_{up}^{(k)} &= \min\left(l^{(k)}\,,\; l_{up}^{(k+1)} + e_{3t}^{(k)}\right) && \text{from } k = 1 \text{ to } jpk \\ l_{dwn}^{(k)} &= \min\left(l^{(k)}\,,\; l_{dwn}^{(k-1)} + e_{3t}^{(k-1)}\right) && \text{from } k = jpk \text{ to } 1 \end{aligned} \tag{10.6}$$

where $l^{(k)}$ is computed using (10.4), *i.e.* $l^{(k)} = \sqrt{2\bar{e}^{(k)}/N^{2(k)}}$.

In the *nn_mxl* = 2 case, the dissipation and mixing length scales take the same value : $l_k = l_\epsilon = \min(\, l_{up}\,,\, l_{dwn}\,)$, while in the *nn_mxl* = 3 case, the dissipation and mixing turbulent length scales are give as in **?** :

$$\begin{aligned} l_k &= \sqrt{l_{up}\;\, l_{dwn}} \\ l_\epsilon &= \min(\, l_{up}\,,\, l_{dwn}\,) \end{aligned} \tag{10.7}$$

At the ocean surface, a non zero length scale is set through the *rn_lmin0* namelist parameter. Usually the surface scale is given by $l_o = \kappa\, z_o$ where $\kappa = 0.4$ is von Karman's constant and z_o the roughness parameter of the surface. Assuming $z_o = 0.1$ m [**?**] leads to a 0.04 m, the default value of *rn_lsurf*. In the ocean interior a minimum length scale is set to recover the molecular viscosity when $\bar{e}$ reach its minimum value ($1.10^{-6} = C_k\, l_{min}\, \sqrt{\bar{e}_{min}}$).

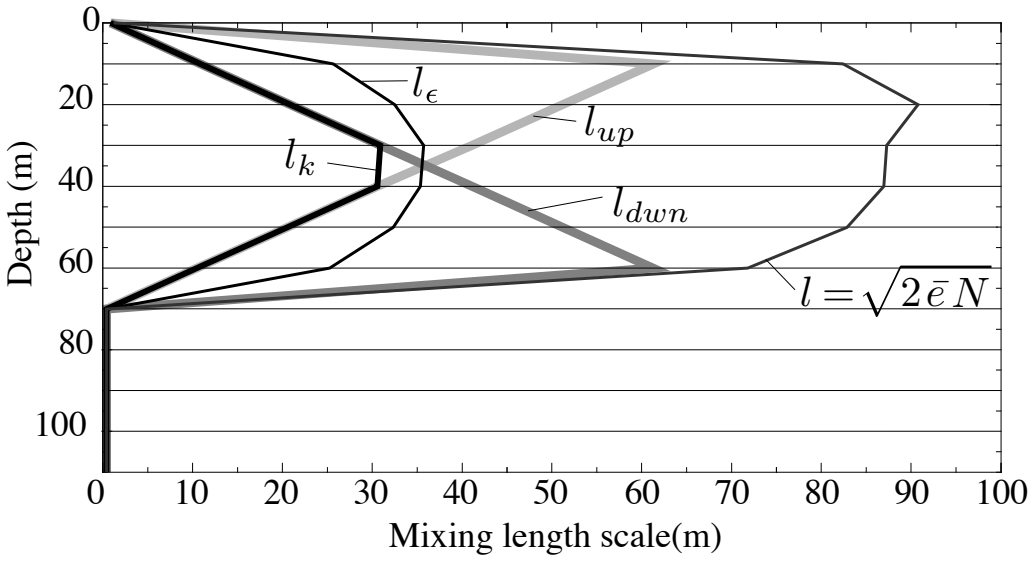

FIG. 10.1 – Illustration of the mixing length computation.

Surface wave breaking parameterization

Following **?**, the TKE turbulence closure model has been modified to include the effect of surface wave breaking energetics. This results in a reduction of summertime surface temperature when the mixed layer is relatively shallow. The **?** modifications acts on surface length scale and TKE values and air-sea drag coefficient. The latter concerns the bulk formulea and is not discussed here.

Following **?**, the boundary condition on surface TKE value is :

$$\bar{e}_o = \frac{1}{2}\,(15.8\,\alpha_{CB})^{2/3}\,\frac{|\tau|}{\rho_o} \tag{10.8}$$

where α_{CB} is the **?** constant of proportionality which depends on the "wave age", ranging from 57 for mature waves to 146 for younger waves [**?**]. The boundary condition on the turbulent length scale follows the Charnock's relation :

$$l_o = \kappa\beta\,\frac{|\tau|}{g\,\rho_o} \tag{10.9}$$

where $\kappa = 0.40$ is the von Karman constant, and β is the Charnock's constant. **?** suggest $\beta = 2.10^5$ the value chosen by **?** citing observation evidence, and $\alpha_{CB} = 100$ the Craig and Banner's value. As the surface boundary condition on TKE is prescribed through $\bar{e}_o = e_{bb}|\tau|/\rho_o$, with e_{bb} the *rn_ebb* namelist parameter, setting *rn_ebb* = 67.83 corresponds to

$\alpha_{CB} = 100$. further setting *ln_lsurf* to true applies (10.9) as surface boundary condition on length scale, with β hard coded to the Stacet's value. Note that a minimal threshold of *rn_emin0*$= 10^{-4}\ m^2.s^{-2}$ (namelist parameters) is applied on surface $\bar{e}$ value.

Langmuir cells

Langmuir circulations (LC) can be described as ordered large-scale vertical motions in the surface layer of the oceans. Although LC have nothing to do with convection, the circulation pattern is rather similar to so-called convective rolls in the atmospheric boundary layer. The detailed physics behind LC is described in, for example, **?**. The prevailing explanation is that LC arise from a nonlinear interaction between the Stokes drift and wind drift currents.

Here we introduced in the TKE turbulent closure the simple parameterization of Langmuir circulations proposed by [**?**] for a $k-\epsilon$ turbulent closure. The parameterization, tuned against large-eddy simulation, includes the whole effect of LC in an extra source terms of TKE, P_{LC}. The presence of P_{LC} in (10.2), the TKE equation, is controlled by setting *ln_lc* to *true* in the namtke namelist.

By making an analogy with the characteristic convective velocity scale ($e.g.$, **?**), P_{LC} is assumed to be :

$$P_{LC}(z) = \frac{w_{LC}^3(z)}{H_{LC}} \tag{10.10}$$

where $w_{LC}(z)$ is the vertical velocity profile of LC, and H_{LC} is the LC depth. With no information about the wave field, w_{LC} is assumed to be proportional to the Stokes drift $u_s = 0.377\ |\tau|^{1/2}$, where $|\tau|$ is the surface wind stress module [1]. For the vertical variation, w_{LC} is assumed to be zero at the surface as well as at a finite depth H_{LC} (which is often close to the mixed layer depth), and simply varies as a sine function in between (a first-order profile for the Langmuir cell structures). The resulting expression for w_{LC} is :

$$w_{LC} = \begin{cases} c_{LC}\, u_s\, \sin(-\pi\, z / H_{LC}) & \text{if } -z \leq H_{LC} \\ 0 & \text{otherwise} \end{cases} \tag{10.11}$$

where $c_{LC} = 0.15$ has been chosen by [**?**] as a good compromise to fit LES data. The chosen value yields maximum vertical velocities w_{LC} of the order of a few centimeters per second. The value of c_{LC} is set through the *rn_lc* namelist parameter, having in mind that it should stay between 0.15 and 0.54 [**?**].

The H_{LC} is estimated in a similar way as the turbulent length scale of TKE equations : H_{LC} is depth to which a water parcel with kinetic energy due to Stoke drift can reach on its own by converting its kinetic energy to potential energy, according to

$$-\int_{-H_{LC}}^{0} N^2\ z\ dz = \frac{1}{2} u_s^2 \tag{10.12}$$

[1] Following **?**, the surface Stoke drift velocity may be expressed as $u_s = 0.016\,|U_{10m}|$. Assuming an air density of $\rho_a = 1.22\,Kg/m^3$ and a drag coefficient of $1.5\ 10^{-3}$ give the expression used of u_s as a function of the module of surface stress

Mixing just below the mixed layer

To be add here a description of "penetration of TKE" and the associated namelist parameters *nn_etau*, *rn_efr* and *nn_htau*.

10.1.4 TKE discretization considerations (key_zdftke)

The production of turbulence by vertical shear (the first term of the right hand side of (10.2)) should balance the loss of kinetic energy associated with the vertical momentum diffusion (first line in (2.36)). To do so a special care have to be taken for both the time and space discretization of the TKE equation [**??**].

Let us first address the time stepping issue. Fig. 10.2 shows how the two-level Leap-Frog time stepping of the momentum and tracer equations interplays with the one-level forward time stepping of TKE equation. With this framework, the total loss of kinetic energy (in 1D for the demonstration) due to the vertical momentum diffusion is obtained by multiplying this quantity by u^t and summing the result vertically :

$$\begin{aligned} &\int_{-H}^{\eta} u^t \, \partial_z \left({K_m}^t \, (\partial_z u)^{t+\Delta t} \right) \, dz \\ &\qquad = \left[u^t \, {K_m}^t \, (\partial_z u)^{t+\Delta t} \right]_{-H}^{\eta} - \int_{-H}^{\eta} {K_m}^t \, \partial_z u^t \, \partial_z u^{t+\Delta t} \, dz \end{aligned} \tag{10.13}$$

Here, the vertical diffusion of momentum is discretized backward in time with a coefficient, K_m, known at time t (Fig. 10.2), as it is required when using the TKE scheme (see §3.3). The first term of the right hand side of (10.13) represents the kinetic energy transfer at the surface (atmospheric forcing) and at the bottom (friction effect). The second term is always negative. It is the dissipation rate of kinetic energy, and thus minus the shear production rate of $\bar{e}$. (10.13) implies that, to be energetically consistent, the production rate of $\bar{e}$ used to compute $(\bar{e})^t$ (and thus ${K_m}^t$) should be expressed as ${K_m}^{t-\Delta t} \, (\partial_z u)^{t-\Delta t} \, (\partial_z u)^t$ (and not by the more straightforward $K_m \, (\partial_z u)^2$ expression taken at time t or $t - \Delta t$).

A similar consideration applies on the destruction rate of $\bar{e}$ due to stratification (second term of the right hand side of (10.2)). This term must balance the input of potential energy resulting from vertical mixing. The rate of change of potential energy (in 1D for the demonstration) due vertical mixing is obtained by multiplying vertical density diffusion tendency by $g \, z$ and and summing the result vertically :

$$\begin{aligned} &\int_{-H}^{\eta} g \, z \, \partial_z \left({K_\rho}^t \, (\partial_k \rho)^{t+\Delta t} \right) \, dz \\ &\qquad = \left[g \, z \, {K_\rho}^t \, (\partial_z \rho)^{t+\Delta t} \right]_{-H}^{\eta} - \int_{-H}^{\eta} g \, {K_\rho}^t \, (\partial_k \rho)^{t+\Delta t} \, dz \\ &\qquad = - \left[z \, {K_\rho}^t \, (N^2)^{t+\Delta t} \right]_{-H}^{\eta} + \int_{-H}^{\eta} \rho^{t+\Delta t} \, {K_\rho}^t \, (N^2)^{t+\Delta t} \, dz \end{aligned} \tag{10.14}$$

where we use $N^2 = -g \, \partial_k \rho / (e_3 \rho)$. The first term of the right hand side of (10.14) is always zero because there is no diffusive flux through the ocean surface and bottom).

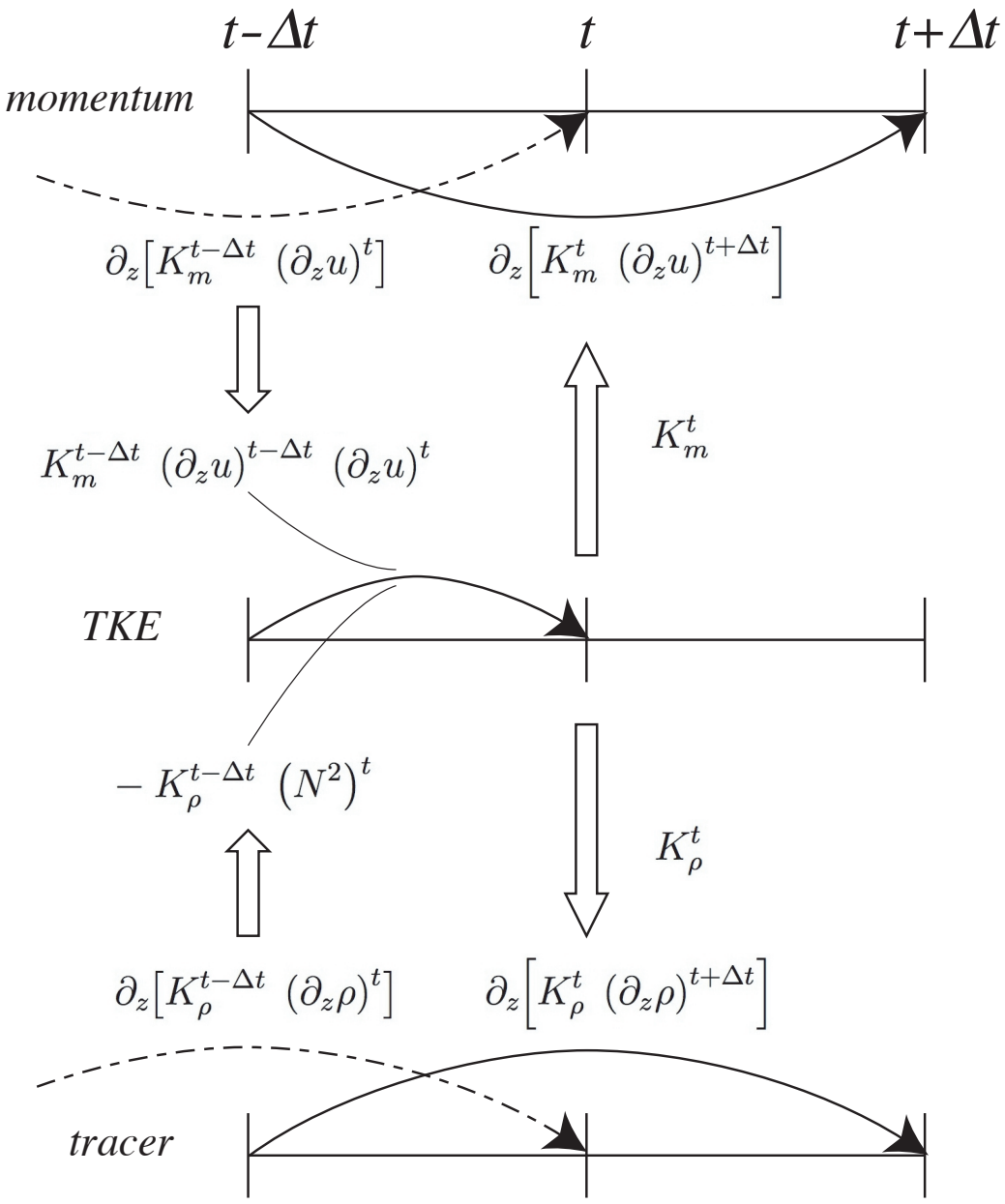

FIG. 10.2 – Illustration of the TKE time integration and its links to the momentum and tracer time integration.

The second term is minus the destruction rate of $\bar{e}$ due to stratification. Therefore (10.13) implies that, to be energetically consistent, the product ${K_\rho}^{t-\Delta t}\,(N^2)^t$ should be used in (10.2), the TKE equation.

Let us now address the space discretization issue. The vertical eddy coefficients are defined at w-point whereas the horizontal velocity components are in the centre of the side faces of a t-box in staggered C-grid (Fig.4.1). A space averaging is thus required to obtain the shear TKE production term. By redoing the (10.13) in the 3D case, it can be shown that the product of eddy coefficient by the shear at t and $t-\Delta t$ must be performed prior to the averaging. Furthermore, the possible time variation of e_3 (**key_vvl** case) have to be taken into account.

The above energetic considerations leads to the following final discrete form for the TKE equation :

$$
\begin{aligned}
\frac{(\bar{e})^t - (\bar{e})^{t-\Delta t}}{\Delta t} \equiv & \left\{ \overline{\left(\left(\overline{K_m}^{i+1/2} \right)^{t-\Delta t} \frac{\delta_{k+1/2}[u^{t+\Delta t}]}{e_3 u^{t+\Delta t}} \frac{\delta_{k+1/2}[u^t]}{e_3 u^t} \right)}^{i} \right. \\
& \left. + \overline{\left(\left(\overline{K_m}^{j+1/2} \right)^{t-\Delta t} \frac{\delta_{k+1/2}[v^{t+\Delta t}]}{e_3 v^{t+\Delta t}} \frac{\delta_{k+1/2}[v^t]}{e_3 v^t} \right)}^{j} \right\} \\
& - {K_\rho}^{t-\Delta t} (N^2)^t \\
& + \frac{1}{e_3 w^{t+\Delta t}} \delta_{k+1/2} \left[{K_m}^{t-\Delta t} \frac{\delta_k[(\bar{e})^{t+\Delta t}]}{e_3 w^{t+\Delta t}} \right] \\
& - c_\epsilon \left(\frac{\sqrt{\bar{e}}}{l_\epsilon} \right)^{t-\Delta t} (\bar{e})^{t+\Delta t}
\end{aligned}
\tag{10.15}
$$

where the last two terms in (10.15) (vertical diffusion and Kolmogorov dissipation) are time stepped using a backward scheme (see§3.3). Note that the Kolmogorov term has been linearized in time in order to render the implicit computation possible. The restart of the TKE scheme requires the storage of $\bar{e}$, K_m, K_ρ and l_ϵ as they all appear in the right hand side of (10.15). For the latter, it is in fact the ratio $\sqrt{\bar{e}}/l_\epsilon$ which is stored.

10.1.5 GLS Generic Length Scale (key_zdfgls)

```
!-----------------------------------------------------------------------
&namzdf_gls                !   GLS vertical diffusion                  ("key_zdfgls")
!-----------------------------------------------------------------------
   rn_emin       = 1.e-6   !  minimum value of e   [m2/s2]
   rn_epsmin     = 1.e-12  !  minimum value of eps [m2/s3]
   ln_length_lim = .true.  !  limit on the dissipation rate under stable stratification (Galperin et al., 1988)
   rn_clim_galp  = 0.53    !  galperin limit
   ln_crban      = .true.  !  Use Craig & Banner (1994) surface wave mixing parametrisation
   ln_sigpsi     = .true.  !  Activate or not Burchard 2001 mods on psi schmidt number in the wb case
   rn_crban      = 100.    !  Craig and Banner 1994 constant for wb tke flux
   rn_charn      = 70000.  !  Charnock constant for wb induced roughness length
   nn_tkebc_surf =   1     !  surface tke condition (0/1/2=Dir/Neum/Dir Mellor-Blumberg)
   nn_tkebc_bot  =   1     !  bottom tke condition (0/1=Dir/Neum)
   nn_psibc_surf =   1     !  surface psi condition (0/1/2=Dir/Neum/Dir Mellor-Blumberg)
   nn_psibc_bot  =   1     !  bottom psi condition (0/1=Dir/Neum)
   nn_stab_func  =   2     !  stability function (0=Galp, 1= KC94, 2=CanutoA, 3=CanutoB)
   nn_clos       =   1     !  predefined closure type (0=MY82, 1=k-eps, 2=k-w, 3=Gen)
/
```

The Generic Length Scale (GLS) scheme is a turbulent closure scheme based on two prognostic equations : one for the turbulent kinetic energy $\bar{e}$, and another for the generic length scale, ψ [**??**]. This later variable is defined as : $\psi = {C_{0\mu}}^p \, \bar{e}^m \, l^n$, where the triplet (p, m, n) value given in Tab.10.1 allows to recover a number of well-known turbulent

closures (k-kl [**?**], k-ϵ [**?**], k-ω [**?**] among others [**??**]). The GLS scheme is given by the following set of equations :

$$\frac{\partial \bar{e}}{\partial t} = \frac{K_m}{\sigma_e e_3} \left[\left(\frac{\partial u}{\partial k} \right)^2 + \left(\frac{\partial v}{\partial k} \right)^2 \right] - K_\rho \, N^2 + \frac{1}{e_3} \frac{\partial}{\partial k} \left[\frac{K_m}{e_3} \frac{\partial \bar{e}}{\partial k} \right] - \epsilon \tag{10.16}$$

$$\begin{aligned} \frac{\partial \psi}{\partial t} =& \frac{\psi}{\bar{e}} \left\{ \frac{C_1 \, K_m}{\sigma_\psi e_3} \left[\left(\frac{\partial u}{\partial k} \right)^2 + \left(\frac{\partial v}{\partial k} \right)^2 \right] - C_3 \, K_\rho \, N^2 - C_2 \, \epsilon \, Fw \right\} \\ &+ \frac{1}{e_3} \frac{\partial}{\partial k} \left[\frac{K_m}{e_3} \frac{\partial \psi}{\partial k} \right] \end{aligned} \tag{10.17}$$

$$\begin{aligned} K_m &= C_\mu \sqrt{\bar{e}} \, l \\ K_\rho &= C_{\mu'} \sqrt{\bar{e}} \, l \end{aligned} \tag{10.18}$$

$$\epsilon = C_{0\mu} \frac{\bar{e}^{3/2}}{l} \tag{10.19}$$

where N is the local Brunt-Vaisälä frequency (see §5.8.2) and ϵ the dissipation rate. The constants C_1, C_2, C_3, σ_e, σ_ψ and the wall function (Fw) depends of the choice of the turbulence model. Four different turbulent models are pre-defined (Tab.10.1). They are made available through the *nn_clo* namelist parameter.

	$k-kl$	$k-\epsilon$	$k-\omega$	generic
nn_clo	**0**	**1**	**2**	**3**
(p,n,m)	(0 , 1 , 1)	(3 , 1.5 , -1)	(-1 , 0.5 , -1)	(2 , 1 , -0.67)
σ_k	2.44	1.	2.	0.8
σ_ψ	2.44	1.3	2.	1.07
C_1	0.9	1.44	0.555	1.
C_2	0.5	1.92	0.833	1.22
C_3	1.	1.	1.	1.
F_{wall}	Yes	–	–	–

TAB. 10.1 – Set of predefined GLS parameters, or equivalently predefined turbulence models available with **key_zdfgls** and controlled by the *nn_clos* namelist parameter.

In the Mellor-Yamada model, the negativity of n allows to use a wall function to force the convergence of the mixing length towards Kz_b (K : Kappa and z_b : rugosity length) value near physical boundaries (logarithmic boundary layer law). C_μ and $C_{\mu'}$ are calculated from stability function proposed by **?**, or by **?** or one of the two functions

suggested by **?** (*nn_stab_func* = 0, 1, 2 or 3, resp.). The value of $C_{0\mu}$ depends of the choice of the stability function.

The surface and bottom boundary condition on both $\bar{e}$ and ψ can be calculated thanks to Dirichlet or Neumann condition through *nn_tkebc_surf* and *nn_tkebc_bot*, resp. As for TKE closure , the wave effect on the mixing is considered when *ln_crban* = true [**??**]. The *rn_crban* namelist parameter is α_{CB} in (10.8) and *rn_charn* provides the value of β in (10.9).

The ψ equation is known to fail in stably stratified flows, and for this reason almost all authors apply a clipping of the length scale as an *ad hoc* remedy. With this clipping, the maximum permissible length scale is determined by $l_{max} = c_{lim}\sqrt{2\bar{e}}/N$. A value of $c_{lim} = 0.53$ is often used [**?**]. **?** show that the value of the clipping factor is of crucial importance for the entrainment depth predicted in stably stratified situations, and that its value has to be chosen in accordance with the algebraic model for the turbulent ßuxes. The clipping is only activated if *ln_length_lim*=true, and the c_{lim} is set to the *rn_clim_galp* value.

The time and space discretization of the GLS equations follows the same energetic consideration as for the TKE case described in §10.1.4 [**?**]. Examples of performance of the 4 turbulent closure scheme can be found in **?**.

10.1.6 K Profile Parametrisation (KPP) (key_zdfkpp)

```
!-----------------------------------------------------------------------
&namzdf_kpp    !   K-Profile Parameterization dependent vertical mixing  ("key_zdfkpp", and optionally:
!------------------------------------------------------------------------ "key_kppcustom" or "key_kpplktb")
   ln_kpprimix = .true.    !  shear instability mixing
   rn_difmiw   =  1.0e-04  !  constant internal wave viscosity [m2/s]
   rn_difsiw   =  0.1e-04  !  constant internal wave diffusivity [m2/s]
   rn_riinfty  =  0.8      !  local Richardson Number limit for shear instability
   rn_difri    =  0.0050   !  maximum shear mixing at Rig = 0    [m2/s]
   rn_bvsqcon  = -0.01e-07 !  Brunt-Vaisala squared for maximum convection [1/s2]
   rn_difcon   =  1.       !  maximum mixing in interior convection [m2/s]
   nn_avb      =  0        !  horizontal averaged (=1) or not (=0) on avt and amv
   nn_ave      =  1        !  constant (=0) or profile (=1) background on avt
/
```

The KKP scheme has been implemented by J. Chanut ...

Add a description of KPP here.

10.2 Convection

```
!-----------------------------------------------------------------------
&namzdf        !   vertical physics
!-----------------------------------------------------------------------
   rn_avm0     =   1.2e-4  !  vertical eddy viscosity   [m2/s]          (background Kz if not "key_zdfcst")
   rn_avt0     =   1.2e-5  !  vertical eddy diffusivity [m2/s]          (background Kz if not "key_zdfcst")
   nn_avb      =    0      !  profile for background avt & avm (=1) or not (=0)
   nn_havtb    =    0      !  horizontal shape for avtb (=1) or not (=0)
   ln_zdfevd   = .true.    !  enhanced vertical diffusion (evd) (T) or not (F)
   nn_evdm     =    0      !  evd apply on tracer (=0) or on tracer and momentum (=1)
   rn_avevd    =  100.     !  evd mixing coefficient [m2/s]
   ln_zdfnpc   = .false.   !  Non-Penetrative Convective algorithm (T) or not (F)
   nn_npc      =    1           !  frequency of application of npc
   nn_npcp     =  365           !  npc control print frequency
   ln_zdfexp   = .false.   !  time-stepping: split-explicit (T) or implicit (F) time stepping
   nn_zdfexp   =    3           !  number of sub-timestep for ln_zdfexp=T
/
```

Static instabilities (i.e. light potential densities under heavy ones) may occur at particular ocean grid points. In nature, convective processes quickly re-establish the static

stability of the water column. These processes have been removed from the model via the hydrostatic assumption so they must be parameterized. Three parameterisations are available to deal with convective processes : a non-penetrative convective adjustment or an enhanced vertical diffusion, or/and the use of a turbulent closure scheme.

10.2.1 Non-Penetrative Convective Adjustment (*ln_tranpc*=.true.)

```
!-----------------------------------------------------------------------
&namzdf        !   vertical physics
!-----------------------------------------------------------------------
   rn_avm0     =   1.2e-4  !  vertical eddy viscosity   [m2/s]          (background Kz if not "key_zdfcst")
   rn_avt0     =   1.2e-5  !  vertical eddy diffusivity [m2/s]          (background Kz if not "key_zdfcst")
   nn_avb      =    0      !  profile for background avt & avm (=1) or not (=0)
   nn_havtb    =    0      !  horizontal shape for avtb (=1) or not (=0)
   ln_zdfevd   = .true.    !  enhanced vertical diffusion (evd) (T) or not (F)
   nn_evdm     =    0      !  evd apply on tracer (=0) or on tracer and momentum (=1)
   rn_avevd    =  100.     !  evd mixing coefficient [m2/s]
   ln_zdfnpc   = .false.   !  Non-Penetrative Convective algorithm (T) or not (F)
   nn_npc      =    1           !  frequency of application of npc
   nn_npcp     =  365           !  npc control print frequency
   ln_zdfexp   = .false.   !  time-stepping: split-explicit (T) or implicit (F) time stepping
   nn_zdfexp   =    3           !  number of sub-timestep for ln_zdfexp=T
/
```

The non-penetrative convective adjustment is used when *ln_zdfnpc*=true. It is applied at each *nn_npc* time step and mixes downwards instantaneously the statically unstable portion of the water column, but only until the density structure becomes neutrally stable ($i.e.$ until the mixed portion of the water column has *exactly* the density of the water just below) [**?**]. The associated algorithm is an iterative process used in the following way (Fig. 10.3) : starting from the top of the ocean, the first instability is found. Assume in the following that the instability is located between levels k and $k+1$. The potential temperature and salinity in the two levels are vertically mixed, conserving the heat and salt contents of the water column. The new density is then computed by a linear approximation. If the new density profile is still unstable between levels $k+1$ and $k+2$, levels k, $k+1$ and $k+2$ are then mixed. This process is repeated until stability is established below the level k (the mixing process can go down to the ocean bottom). The algorithm is repeated to check if the density profile between level $k-1$ and k is unstable and/or if there is no deeper instability.

This algorithm is significantly different from mixing statically unstable levels two by two. The latter procedure cannot converge with a finite number of iterations for some vertical profiles while the algorithm used in *NEMO* converges for any profile in a number of iterations which is less than the number of vertical levels. This property is of paramount importance as pointed out by **?** : it avoids the existence of permanent and unrealistic static instabilities at the sea surface. This non-penetrative convective algorithm has been proved successful in studies of the deep water formation in the north-western Mediterranean Sea [**???**].

Note that in the current implementation of this algorithm presents several limitations. First, potential density referenced to the sea surface is used to check whether the density profile is stable or not. This is a strong simplification which leads to large errors for realistic ocean simulations. Indeed, many water masses of the world ocean, especially Antarctic Bottom Water, are unstable when represented in surface-referenced potential

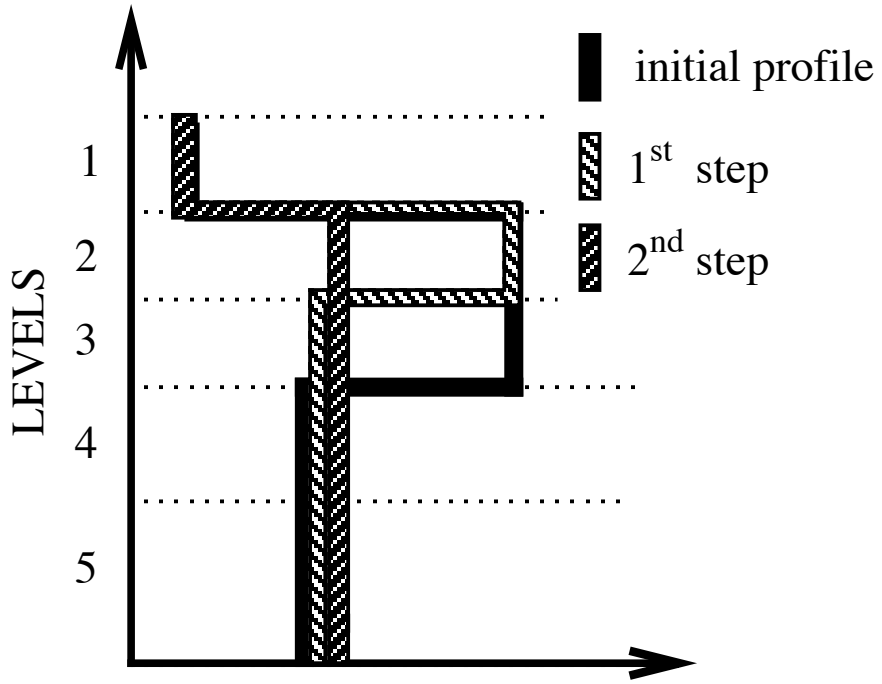

FIG. 10.3 – Example of an unstable density profile treated by the non penetrative convective adjustment algorithm. 1^{st} step : the initial profile is checked from the surface to the bottom. It is found to be unstable between levels 3 and 4. They are mixed. The resulting ρ is still larger than $\rho(5)$: 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.

density. The scheme will erroneously mix them up. Second, the mixing of potential density is assumed to be linear. This assures the convergence of the algorithm even when the equation of state is non-linear. Small static instabilities can thus persist due to cabbeling : they will be treated at the next time step. Third, temperature and salinity, and thus density, are mixed, but the corresponding velocity fields remain unchanged. When using a Richardson Number dependent eddy viscosity, the mixing of momentum is done through the vertical diffusion : after a static adjustment, the Richardson Number is zero and thus the eddy viscosity coefficient is at a maximum. When this convective adjustment algorithm is used with constant vertical eddy viscosity, spurious solutions can occur since the vertical momentum diffusion remains small even after a static adjustment. In that case, we recommend the addition of momentum mixing in a manner that mimics the mixing in temperature and salinity [**??**].

10.2.2 Enhanced Vertical Diffusion (*ln_zdfevd*=true)

```
!-----------------------------------------------------------------------
&namzdf        !   vertical physics
!-----------------------------------------------------------------------
   rn_avm0     =   1.2e-4  !  vertical eddy viscosity   [m2/s]          (background Kz if not "key_zdfcst")
   rn_avt0     =   1.2e-5  !  vertical eddy diffusivity [m2/s]          (background Kz if not "key_zdfcst")
   nn_avb      =    0      !  profile for background avt & avm (=1) or not (=0)
   nn_havtb    =    0      !  horizontal shape for avtb (=1) or not (=0)
   ln_zdfevd   = .true.    !  enhanced vertical diffusion (evd) (T) or not (F)
   nn_evdm     =    0      !  evd apply on tracer (=0) or on tracer and momentum (=1)
   rn_avevd    =  100.     !  evd mixing coefficient [m2/s]
   ln_zdfnpc   = .false.   !  Non-Penetrative Convective algorithm (T) or not (F)
   nn_npc      =    1           !  frequency of application of npc
   nn_npcp     =  365           !  npc control print frequency
   ln_zdfexp   = .false.   !  time-stepping: split-explicit (T) or implicit (F) time stepping
   nn_zdfexp   =    3           !  number of sub-timestep for ln_zdfexp=T
/
```

The enhanced vertical diffusion parameterisation is used when *ln_zdfevd*=true. In this case, the vertical eddy mixing coefficients are assigned very large values (a typical value is 10 $m^2 s^{-1}$) in regions where the stratification is unstable ($i.e.$ when N^2 the Brunt-Vaisälä frequency is negative) [**??**]. This is done either on tracers only (*nn_evdm*=0) or on both momentum and tracers (*nn_evdm*=1).

In practice, where $N^2 \leq 10^{-12}$, A_T^{vT} and A_T^{vS}, and if *nn_evdm*=1, the four neighbouring A_u^{vm} and A_v^{vm} values also, are set equal to the namelist parameter *rn_avevd*. A typical value for rn_avevd is between 1 and 100 $m^2.s^{-1}$. This parameterisation of convective processes is less time consuming than the convective adjustment algorithm presented above when mixing both tracers and momentum in the case of static instabilities. It requires the use of an implicit time stepping on vertical diffusion terms (i.e. *ln_zdfexp*=false).

Note that the stability test is performed on both *before* and *now* values of N^2. This removes a potential source of divergence of odd and even time step in a leapfrog environment [**?**] (see §3.5).

10.2.3 Turbulent Closure Scheme (key_zdftke or key_zdfgls)

The turbulent closure scheme presented in §10.1.3 and §10.1.5 (**key_zdftke** or **key_zdftke** is defined) in theory solves the problem of statically unstable density profiles. In such a case, the term corresponding to the destruction of turbulent kinetic energy through stratification in (10.2) or (10.16) becomes a source term, since N^2 is negative. It results in large values of A_T^{vT} and A_T^{vT}, and also the four neighbouring $A_u^{vm} and\ A_v^{vm}$ (up to 1 $m^2 s^{-1}$). These large values restore the static stability of the water column in a way similar to that of the enhanced vertical diffusion parameterisation (§10.2.2). However, in the vicinity of the sea surface (first ocean layer), the eddy coefficients computed by the turbulent closure scheme do not usually exceed $10^{-2} m.s^{-1}$, because the mixing length scale is bounded by the distance to the sea surface. It can thus be useful to combine the enhanced vertical diffusion with the turbulent closure scheme, $i.e.$ setting the *ln_zdfnpc* namelist parameter to true and defining the turbulent closure CPP key all together.

The KPP turbulent closure scheme already includes enhanced vertical diffusion in the case of convection, as governed by the variables $bvsqcon$ and $difcon$ found in *zdfkpp.F90*, therefore *ln_zdfevd*=false should be used with the KPP scheme.

10.3 Double Diffusion Mixing (key_zdfddm)

```
!-----------------------------------------------------------------------
&namzdf_ddm    !   double diffusive mixing parameterization            ("key_zdfddm")
!-----------------------------------------------------------------------
   rn_avts     = 1.e-4     !  maximum avs (vertical mixing on salinity)
   rn_hsbfr    = 1.6       !  heat/salt buoyancy flux ratio
/
```

Double diffusion occurs when relatively warm, salty water overlies cooler, fresher water, or vice versa. The former condition leads to salt fingering and the latter to diffusive convection. Double-diffusive phenomena contribute to diapycnal mixing in extensive regions of the ocean. **?** include a parameterisation of such phenomena in a global ocean model and show that it leads to relatively minor changes in circulation but exerts significant regional influences on temperature and salinity. This parameterisation has been introduced in *zdfddm.F90* module and is controlled by the **key_zdfddm** CPP key.

Diapycnal mixing of S and T are described by diapycnal diffusion coefficients

$$A^{vT} = A_o^{vT} + A_f^{vT} + A_d^{vT}$$
$$A^{vS} = A_o^{vS} + A_f^{vS} + A_d^{vS}$$

where subscript f represents mixing by salt fingering, d by diffusive convection, and o by processes other than double diffusion. The rates of double-diffusive mixing depend on the buoyancy ratio $R_\rho = \alpha \partial_z T / \beta \partial_z S$, where α and β are coefficients of thermal expansion and saline contraction (see §5.8.1). To represent mixing of S and T by salt fingering, we adopt the diapycnal diffusivities suggested by Schmitt (1981) :

$$A_f^{vS} = \begin{cases} \frac{A^{*v}}{1+(R_\rho/R_c)^n} & \text{if } R_\rho > 1 \text{ and } N^2 > 0 \\ 0 & \text{otherwise} \end{cases} \tag{10.20}$$

$$A_f^{vT} = 0.7 \; A_f^{vS} / R_\rho \tag{10.21}$$

The factor 0.7 in (10.21) reflects the measured ratio $\alpha F_T / \beta F_S \approx 0.7$ of buoyancy flux of heat to buoyancy flux of salt (*e.g.*, **?**). Following **?**, we adopt $R_c = 1.6$, $n = 6$, and $A^{*v} = 10^{-4} \; m^2.s^{-1}$.

To represent mixing of S and T by diffusive layering, the diapycnal diffusivities suggested by Federov (1988) is used :

$$A_d^{vT} = \begin{cases} 1.3635 \, \exp\left(4.6 \, \exp\left[-0.54 \, (R_\rho^{-1} - 1)\right]\right) & \text{if } 0 < R_\rho < 1 \text{ and } N^2 > 0 \\ 0 & \text{otherwise} \end{cases} \tag{10.22}$$

$$A_d^{vS} = \begin{cases} A_d^{vT} \; (1.85 \, R_\rho - 0.85) & \text{if } 0.5 \leq R_\rho < 1 \text{ and } N^2 > 0 \\ A_d^{vT} \; 0.15 \; R_\rho & \text{if } \; 0 < R_\rho < 0.5 \text{ and } N^2 > 0 \\ 0 & \text{otherwise} \end{cases} \tag{10.23}$$

The dependencies of (10.20) to (10.23) on R_ρ are illustrated in Fig. 10.4. Implementing this requires computing R_ρ at each grid point on every time step. This is done in *eosbn2.F90* at the same time as N^2 is computed. This avoids duplication in the computation of α and β (which is usually quite expensive).

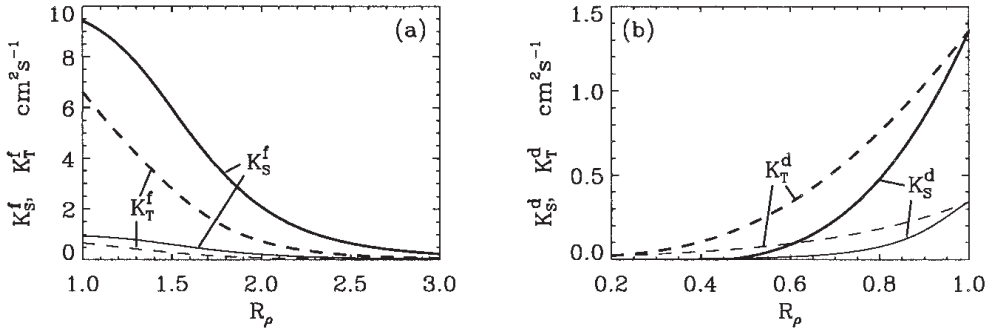

FIG. 10.4 – From **?** : (a) Diapycnal diffusivities A_f^{vT} and A_f^{vS} for temperature and salt in regions of salt fingering. Heavy curves denote $A^{*v} = 10^{-3}\ m^2.s^{-1}$ and thin curves $A^{*v} = 10^{-4}\ m^2.s^{-1}$; (b) diapycnal diffusivities A_d^{vT} and A_d^{vS} for temperature and salt in regions of diffusive convection. Heavy curves denote the Federov parameterisation and thin curves the Kelley parameterisation. The latter is not implemented in *NEMO* .

10.4 Bottom Friction (*zdfbfr.F90* module)

```
!-----------------------------------------------------------------------
&nambfr        !   bottom friction
!-----------------------------------------------------------------------
   nn_bfr      =    1      !  type of bottom friction :   = 0 : free slip,  = 1 : linear friction
                           !                              = 2 : nonlinear friction
   rn_bfri1    =    4.e-4  !  bottom drag coefficient (linear case)
   rn_bfri2    =    1.e-3  !  bottom drag coefficient (non linear case)
   rn_bfeb2    =    2.5e-3 !  bottom turbulent kinetic energy background  (m2/s2)
   ln_bfr2d    = .false.   !  horizontal variation of the bottom friction coef (read a 2D mask file )
   rn_bfrien   =    50.    !  local multiplying factor of bfr (ln_bfr2d=T)
/
```

Both the surface momentum flux (wind stress) and the bottom momentum flux (bottom friction) enter the equations as a condition on the vertical diffusive flux. For the bottom boundary layer, one has :

$$A^{vm}\,(\partial \mathbf{U}_h/\partial z) = \mathcal{F}_h^{\mathbf{U}} \tag{10.24}$$

where $\mathcal{F}_h^{\mathbf{U}}$ is represents the downward flux of horizontal momentum outside the logarithmic turbulent boundary layer (thickness of the order of 1 m in the ocean). How $\mathcal{F}_h^{\mathbf{U}}$ 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 :

$$\frac{\partial u_k}{\partial t} = \frac{1}{e_{3u}} \left[\frac{A^{vm}_{uw}}{e_{3uw}} \delta_{k+1/2} \left[u \right] - \mathcal{F}^u_h \right] \approx - \frac{\mathcal{F}^u_h}{e_{3u}} \tag{10.25}$$

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 $\mathcal{F}^u_h$. Two choices are available in *NEMO* : a linear and a quadratic bottom friction. Note that in both cases, the rotation between the interior velocity and the bottom friction is neglected in the present release of *NEMO* .

In the code, the bottom friction is imposed by adding the trend due to the bottom friction to the general momentum trend in *dynbfr.F90*. For the time-split surface pressure gradient algorithm, the momentum trend due to the barotropic component needs to be handled separately. For this purpose it is convenient to compute and store coefficients which can be simply combined with bottom velocities and geometric values to provide the momentum trend due to bottom friction. These coefficients are computed in *zdfbfr.F90* and generally take the form $c^{\mathbf{U}}_b$ where :

$$\frac{\partial \mathbf{U}_h}{\partial t} = - \frac{\mathcal{F}^{\mathbf{U}}_h}{e_{3u}} = \frac{c^{\mathbf{U}}_b}{e_{3u}} \, \mathbf{U}^b_h \tag{10.26}$$

where $\mathbf{U}^b_h = (u_b \ , \ v_b)$ is the near-bottom, horizontal, ocean velocity.

10.4.1 Linear Bottom Friction (*nn_botfr* = 0 or 1)

The linear bottom friction parameterisation (including the special case of a free-slip condition) assumes that the bottom friction is proportional to the interior velocity (i.e. the velocity of the last model level) :

$$\mathcal{F}^{\mathbf{U}}_h = \frac{A^{vm}}{e_3} \, \frac{\partial \mathbf{U}_h}{\partial k} = r \, \mathbf{U}^b_h \tag{10.27}$$

where r is a friction coefficient expressed in ms^{-1}. This coefficient is generally estimated by setting a typical decay time τ in the deep ocean, and setting $r = H/\tau$, where H is the ocean depth. Commonly accepted values of τ are of the order of 100 to 200 days [**?**]. A value $\tau^{-1} = 10^{-7}$ s^{-1} equivalent to 115 days, is usually used in quasi-geostrophic models. One may consider the linear friction as an approximation of quadratic friction, $r \approx 2 \, C_D \, U_{av}$ (**?**, Eq. 9.6.6). For example, with a drag coefficient $C_D = 0.002$, a typical speed of tidal currents of $U_{av} = 0.1$ m s^{-1}, and assuming an ocean depth $H = 4000$ m, the resulting friction coefficient is $r = 4 \ 10^{-4}$ m s^{-1}. This is the default value used in *NEMO* . It corresponds to a decay time scale of 115 days. It can be changed by specifying *rn_bfric1* (namelist parameter).

For the linear friction case the coefficients defined in the general expression (10.26) are :

$$\begin{aligned} c_b^u &= -r \\ c_b^v &= -r \end{aligned} \tag{10.28}$$

When *nn_botfr*=1, the value of r used is *rn_bfric1*. Setting *nn_botfr*=0 is equivalent to setting $r = 0$ and leads to a free-slip bottom boundary condition. These values are assigned in *zdfbfr.F90*. From v3.2 onwards there is support for local enhancement of these values via an externally defined 2D mask array (*ln_bfr2d*=true) given in the *bfr_coef.nc* input NetCDF file. The mask values should vary from 0 to 1. Locations with a non-zero mask value will have the friction coefficient increased by $mask\_value$**rn_bfrien***rn_bfric1*.

10.4.2 Non-Linear Bottom Friction (*nn_botfr* = 2)

The non-linear bottom friction parameterisation assumes that the bottom friction is quadratic :

$$\mathcal{F}_h^{\mathbf{U}} = \frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} = C_D \sqrt{u_b^2 + v_b^2 + e_b} \ \mathbf{U}_h^b \tag{10.29}$$

where C_D is 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 $e_b = 2.5\ 10^{-3} \text{m}^2\ \text{s}^{-2}$, while the FRAM experiment [**?**] uses $C_D = 1.4\ 10^{-3}$ and $e_b = 2.5\ \ 10^{-3} \text{m}^2\ \text{s}^{-2}$. The CME choices have been set as default values (*rn_bfric2* and *rn_bfeb2* namelist parameters).

As for the linear case, the bottom friction is imposed in the code by adding the trend due to the bottom friction to the general momentum trend in *dynbfr.F90*. For the non-linear friction case the terms computed in *zdfbfr.F90* are :

$$\begin{aligned} c_b^u &= -\ C_D\ \left[u^2 + \left(\bar{\bar{v}}^{i+1,j} \right)^2 + e_b \right]^{1/2} \\ c_b^v &= -\ C_D\ \left[\left(\bar{\bar{u}}^{i,j+1} \right)^2 + v^2 + e_b \right]^{1/2} \end{aligned} \tag{10.30}$$

The coefficients that control the strength of the non-linear bottom friction are initialised as namelist parameters : C_D= *rn_bfri2*, and e_b =*rn_bfeb2*. Note for applications which treat tides explicitly a low or even zero value of *rn_bfeb2* is recommended. From v3.2 onwards a local enhancement of C_D is possible via an externally defined 2D mask array (*ln_bfr2d*=true). See previous section for details.

10.4.3 Bottom Friction stability considerations

Some care needs to exercised over the choice of parameters to ensure that the implementation of bottom friction does not induce numerical instability. For the purposes of

stability analysis, an approximation to (10.25) is :

$$\begin{aligned} \Delta u &= -\frac{{\mathcal{F}_h}^u}{e_{3u}}\, 2\Delta t \\ &= -\frac{ru}{e_{3u}}\, 2\Delta t \end{aligned} \tag{10.31}$$

where linear bottom friction and a leapfrog timestep have been assumed. To ensure that the bottom friction cannot reverse the direction of flow it is necessary to have :

$$|\Delta u| < |u| \tag{10.32}$$

which, using (10.31), gives :

$$r\frac{2\Delta t}{e_{3u}} < 1 \qquad \Rightarrow \qquad r < \frac{e_{3u}}{2\Delta t} \tag{10.33}$$

This same inequality can also be derived in the non-linear bottom friction case if a velocity of 1 m.s^{-1} is assumed. Alternatively, this criterion can be rearranged to suggest a minimum bottom box thickness to ensure stability :

$$e_{3u} > 2\ r\ \Delta t \tag{10.34}$$

which it may be necessary to impose if partial steps are being used. For example, if $|u| = 1$ m.s^{-1}, $rdt = 1800$ s, $r = 10^{-3}$ then e_{3u} should be greater than 3.6 m. For most applications, with physically sensible parameters these restrictions should not be of concern. But caution may be necessary if attempts are made to locally enhance the bottom friction parameters. To ensure stability limits are imposed on the bottom friction coefficients both during initialisation and at each time step. Checks at initialisation are made in *zdfbfr.F90* (assuming a 1 m.s^{-1} velocity in the non-linear case). The number of breaches of the stability criterion are reported as well as the minimum and maximum values that have been set. The criterion is also checked at each time step, using the actual velocity, in *dynbfr.F90*. Values of the bottom friction coefficient are reduced as necessary to ensure stability ; these changes are not reported.

10.4.4 Bottom Friction with split-explicit time splitting

When calculating the momentum trend due to bottom friction in *dynbfr.F90*, the bottom velocity at the before time step is used. This velocity includes both the baroclinic and barotropic components which is appropriate when using either the explicit or filtered surface pressure gradient algorithms (**key_dynspg_exp** or **key_dynspg_flt**). Extra attention is required, however, when using split-explicit time stepping (**key_dynspg_ts**). In this case the free surface equation is solved with a small time step *nn_baro*rn_rdt*, while the three dimensional prognostic variables are solved with a longer time step that is a multiple of *rn_rdt*. The trend in the barotropic momentum due to bottom friction appropriate to this method is that given by the selected parameterisation ($i.e.$ linear or non-linear bottom friction) computed with the evolving velocities at each barotropic timestep.

In the case of non-linear bottom friction, we have elected to partially linearise the problem by keeping the coefficients fixed throughout the barotropic time-stepping to those computed in *zdfbfr.F90* using the now timestep. This decision allows an efficient use of the $c_b^{\mathbf{U}}$ coefficients to :

1. On entry to *dyn_spg_ts*, remove the contribution of the before barotropic velocity to the bottom friction component of the vertically integrated momentum trend. Note the same stability check that is carried out on the bottom friction coefficient in *dynbfr.F90* has to be applied here to ensure that the trend removed matches that which was added in *dynbfr.F90*.
2. At each barotropic step, compute the contribution of the current barotropic velocity to the trend due to bottom friction. Add this contribution to the vertically integrated momentum trend. This contribution is handled implicitly which eliminates the need to impose a stability criteria on the values of the bottom friction coefficient within the barotropic loop.

Note that the use of an implicit formulation for the bottom friction trend means that any limiting of the bottom friction coefficient in *dynbfr.F90* does not adversely affect the solution when using split-explicit time splitting. This is because the major contribution to bottom friction is likely to come from the barotropic component which uses the unrestricted value of the coefficient.

The implicit formulation takes the form :

$$\bar{U}^{t+\Delta t} = \left[\bar{U}^{t-\Delta t} + 2\Delta t \; RHS\right] / \left[1 - 2\Delta t \; c_b^u / H_e\right] \tag{10.35}$$

where $\bar{U}$ is the barotropic velocity, H_e is the full depth (including sea surface height), c_b^u is the bottom friction coefficient as calculated in *zdf_bfr* and RHS represents all the components to the vertically integrated momentum trend except for that due to bottom friction.

10.5 Tidal Mixing (key_zdftmx)

```
!-----------------------------------------------------------------------
&namzdf_tmx    !   tidal mixing parameterization                      ("key_zdftmx")
!-----------------------------------------------------------------------
   rn_htmx     = 500.      !  vertical decay scale for turbulence (meters)
   rn_n2min    = 1.e-8     !  threshold of the Brunt-Vaisala frequency (s-1)
   rn_tfe      = 0.333     !  tidal dissipation efficiency
   rn_me       = 0.2       !  mixing efficiency
   ln_tmx_itf  = .true.    !  ITF specific parameterisation
   rn_tfe_itf  = 1.        !  ITF tidal dissipation efficiency
/
```

10.5.1 Bottom intensified tidal mixing

The parameterization of tidal mixing follows the general formulation for the vertical eddy diffusivity proposed by **?** and first introduced in an OGCM by [**?**]. In this formulation

an additional vertical diffusivity resulting from internal tide breaking, A^{vT}_{tides} is expressed as a function of $E(x,y)$, the energy transfer from barotropic tides to baroclinic tides :

$$A^{vT}_{tides} = q\,\Gamma\,\frac{E(x,y)\,F(z)}{\rho\,N^2} \tag{10.36}$$

where Γ is the mixing efficiency, N the Brunt-Vaisälä frequency (see §5.8.2), ρ the density, q the tidal dissipation efficiency, and $F(z)$ the vertical structure function.

The mixing efficiency of turbulence is set by Γ (*rn_me* namelist parameter) and is usually taken to be the canonical value of $\Gamma = 0.2$ (Osborn 1980). The tidal dissipation efficiency is given by the parameter q (*rn_tfe* namelist parameter) represents the part of the internal wave energy flux $E(x,y)$ that is dissipated locally, with the remaining $1-q$ radiating away as low mode internal waves and contributing to the background internal wave field. A value of $q = 1/3$ is typically used **?**. The vertical structure function $F(z)$ models the distribution of the turbulent mixing in the vertical. It is implemented as a simple exponential decaying upward away from the bottom, with a vertical scale of h_o (*rn_htmx* namelist parameter, with a typical value of $500\,m$) [**?**],

$$F(i,j,k) = \frac{e^{-\frac{H+z}{h_o}}}{h_o\left(1-e^{-\frac{H}{h_o}}\right)} \tag{10.37}$$

and is normalized so that vertical integral over the water column is unity.

The associated vertical viscosity is calculated from the vertical diffusivity assuming a Prandtl number of 1, *i.e.* $A^{vm}_{tides} = A^{vT}_{tides}$. In the limit of $N \to 0$ (or becoming negative), the vertical diffusivity is capped at $300\,cm^2/s$ and impose a lower limit on N^2 of *rn_n2min* usually set to $10^{-8}s^{-2}$. These bounds are usually rarely encountered.

The internal wave energy map, $E(x,y)$ in (10.36), is derived from a barotropic model of the tides utilizing a parameterization of the conversion of barotropic tidal energy into internal waves. The essential goal of the parameterization is to represent the momentum exchange between the barotropic tides and the unrepresented internal waves induced by the tidal ßow over rough topography in a stratified ocean. In the current version of *NEMO* , the map is built from the output of the barotropic global ocean tide model MOG2D-G [**?**]. This model provides the dissipation associated with internal wave energy for the M2 and K1 tides component (Fig. 10.5). The S2 dissipation is simply approximated as being $1/4$ of the M2 one. The internal wave energy is thus : $E(x,y) = 1.25E_{M2} + E_{K1}$. Its global mean value is 1.1 TW, in agreement with independent estimates [**??**].

10.5.2 Indonesian area specific treatment (*ln_zdftmx_itf*)

When the Indonesian Through Flow (ITF) area is included in the model domain, a specific treatment of tidal induced mixing in this area can be used. It is activated through the namelist logical *ln_tmx_itf*, and the user must provide an input NetCDF file, *mask_itf.nc*, which contains a mask array defining the ITF area where the specific treatment is applied.

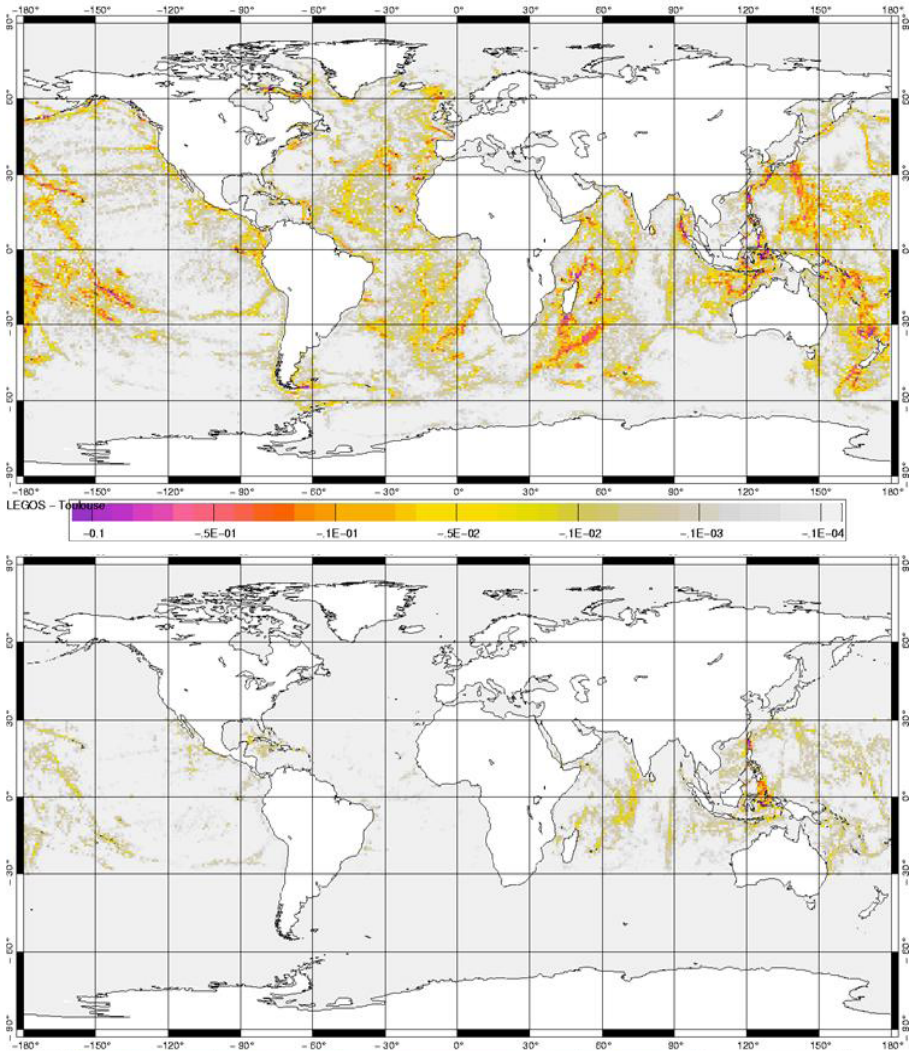

FIG. 10.5 – (a) M2 and (b) K2 internal wave drag energy from **?** (W/m^2).

When *ln_tmx_itf*=true, the two key parameters q and $F(z)$ are adjusted following the parameterisation developed by **??** :

First, the Indonesian archipelago is a complex geographic region with a series of large, deep, semi-enclosed basins connected via numerous narrow straits. Once generated, internal tides remain confined within this semi-enclosed area and hardly radiate away. Therefore all the internal tides energy is consumed within this area. So it is assumed that $q = 1, i.e.$ all the energy generated is available for mixing. Note that for test purposed, the ITF tidal dissipation efficiency is a namelist parameter (*rn_tfe_itf*). A value of 1 or close

to is this recommended for this parameter.

Second, the vertical structure function, $F(z)$, is no more associated with a bottom intensification of the mixing, but with a maximum of energy available within the thermocline. **??** have suggested that the vertical distribution of the energy dissipation proportional to N^2 below the core of the thermocline and to N above. The resulting $F(z)$ is :

$$F(i,j,k) \sim \begin{cases} \dfrac{q\,\Gamma E(i,j)}{\rho N \int N dz} & \text{when } \partial_z N < 0 \\ \dfrac{q\,\Gamma E(i,j)}{\rho \int N^2 dz} & \text{when } \partial_z N > 0 \end{cases} \tag{10.38}$$

Averaged over the ITF area, the resulting tidal mixing coefficient is $1.5\,cm^2/s$, which agrees with the independent estimates inferred from observations. Introduced in a regional OGCM, the parameterization improves the water mass characteristics in the different Indonesian seas, suggesting that the horizontal and vertical distributions of the mixing are adequately prescribed [**???**]. Note also that such a parameterisation has a sugnificant impact on the behaviour of global coupled GCMs [**?**].

11 Ouput and Diagnostics (IOM, DIA, TRD, FLO

Contents

11.1 Old Model Output (default or key_dimgout)

The model outputs are of three types : the restart file, the output listing, and the output file(s). The restart file is used internally by the code when the user wants to start the model with initial conditions defined by a previous simulation. It contains all the information that is necessary in order for there to be no changes in the model results (even at the computer precision) between a run performed with several restarts and the same run performed in one step. It should be noted that this requires that the restart file contain two consecutive time steps for all the prognostic variables, and that it is saved in the same binary format as the one used by the computer that is to read it (in particular, 32 bits binary IEEE format must not be used for this file). The output listing and file(s) are predefined but should be checked and eventually adapted to the user's needs. The output listing is stored in the $ocean.output$ file. The information is printed from within the code on the logical unit $numout$. To locate these prints, use the UNIX command "*grep -i numout*" in the source code directory.

In the standard configuration, the user will find the model results in NetCDF files containing mean values (or instantaneous values if **key_diainstant** is defined) for every time-step where output is demanded. These outputs are defined in the *diawri.F90* module. When defining **key_dimgout**, the output are written in DIMG format, an IEEE output format.

Since version 3.2, an I/O server has been added which provides more flexibility in the choice of the fields to be output as well as how the writing work is distributed over the processors in massively parallel computing. It is presented in next section.

11.2 Standard model Output (IOM)

Since version 3.2, iom_put is the NEMO output interface. It was designed to be simple to use, flexible and efficient. Two main functionalities are covered by iom_put : (1) the control of the output files through an external xml file defined by the user ; (2) the distribution (or not) of all task related to output files on dedicated processors. The first functionality allows the user to specify, without touching anything into the code, the way he want to output data :

- choice of output frequencies that can be different for each file (including real months and years)
- choice of file contents : decide which data will be written in which file (the same data can be outputted in different files)
- possibility to extract a subdomain (for example all TAO-PIRATA-RAMA moorings are already defined)
- choice of the temporal operation to perform : mean, instantaneous, min, max

- extremely large choice of data available
- redefine variables name and long_name
In addition, iom_put allows the user to output any variable (scalar, 2D or 3D) in the code in a very easy way. All details of iom_put functionalities are listed in the following subsections. An example of the iodef.xml file that control the outputs can be found here : NEMOGCM/CONFIG/ORCA2_LIM/EXP00/iodef.xml

The second functionality targets outputs performances when running on a very large number of processes. The idea is to dedicate N specific processes to write the outputs, where N is defined by the user. In the current version, this functionality is technically working however, its performance are usually poor (for known reasons). Users can therefore test this functionality but they must be aware that expected performance improvement will not be achieved before the release 3.4. An example of xmlio_server.def NEMOGCM/CONFIG/ORCA2_LIM/EXP00/xmlio_server.def

11.2.1 Basic knowledge

XML basic rules

XML tags begin with the less-than character ("<") and end with the greater-than character (">"). You use tags to mark the start and end of elements, which are the logical units of information in an XML document. In addition to marking the beginning of an element, XML start tags also provide a place to specify attributes. An attribute specifies a single property for an element, using a name/value pair, for example : ... . See here for more details.

Structure of the xml file used in NEMO

The xml file is split into 3 parts :

field definition : define all variables that can be output
all lines in between the following two tags
```
<field\_definition ...>
</field\_definition ...>
```

file definition : define the netcdf files to be created and the variables they will contain
all lines in between the following two tags
```
<field\_definition>
</field\_definition>
```

axis and grid definitions : define the horizontal and vertical grids
all lines in between the following two set of two tags
```
<axis\_definition ...>
</axis\_definition ...>  and
<grid\_definition ...>
</grid\_definition ...>
```

Inheritance and group

Xml extensively uses the concept of inheritance.

example 1 :

```
<field_definition operation="ave(X)" >
   <field id="sst"                    />   <!-- averaged      sst -->
   <field id="sss" operation="inst(X)"/>   <!-- instantaneous sss -->
</field_definition>
```

The field ”sst” which is part (or a child) of the field_definition will inherit the value ”ave(X)” of the attribute ”operation” from its parent ”field definition”. Note that a child can overwrite the attribute definition inherited from its parents. In the example above, the field ”sss” will therefore output instantaneous values instead of average values.

example 2 : Use (or overwrite) attributes value of a field when listing the variables included in a file

```
<field_definition>
   <field id="sst" description="sea surface temperature" />
   <field id="sss" description="sea surface salinity"    />
</field_definition>

<file_definition>
   <file id="file_1" />
         <field ref="sst"                                /> <!-- default def -->
         <field ref="sss" description="my description" /> <!-- overwrite   -->
   </file>
</file_definition>
```

With the help of the inheritance, the concept of group allow to define a set of attributes for several fields or files.

example 3, group of fields : define a group ”T_grid_variables” identified with the name ”grid_T”. By default variables of this group have no vertical axis but, following inheritance rules, ”axis_ref” can be redefined for the field ”toce” that is a 3D variable.

```
<field_definition>
   <group id="grid_T" axis_ref="none" grid_ref="T_grid_variables">
         <field id="sst"/>
         <field id="sss"/>
         <field id="toce" axis_ref="deptht"/>  <!-- overwrite axis def -->
   </group>
</field_definition>
```

example 4, group of files : define a group of file with the attribute output_freq equal to 432000 (5 days)

```
<file_definition>
   <group id="5d" output_freq="432000">    <!-- 5d files -->
      <file id="5d_grid_T" name="auto">   <!-- T grid file -->
      ...
      </file>
      <file id="5d_grid_U" name="auto">   <!-- U grid file -->
      ...
      </file>
   </group>
</file_definition>
```

Control of the xml attributes from NEMO

The values of some attributes are automatically defined by NEMO (and any definition given in the xml file is overwritten). By convention, these attributes are defined to ”auto” (for string) or ”0000” (for integer) in the xml file (but this is not necessary).

Here is the list of these attributes :

tag ids affected by automatic definition of some of their attributes		name attribute	attribute value
field_definition		freq_op	*rn_rdt*
SBC		freq_op	*rn_rdt* × *nn_fsbc*
1h, 2h, 3h, 4h, 6h, 12h 1d, 3d, 5d 1m, 2m, 3m, 4m, 6m 1y, 2y, 5y, 10y	_grid_T, _grid_U, _grid_V, _grid_W, _icemod, _ptrc_T, _diad_T, _scalar	name	filename defined by a call to roudia_nam following NEMO nomenclature
EqT, EqU, EqW		jbegin, ni, name_suffix	according to the grid
TAO, RAMA and PIRATA moorings		ibegin, jbegin, name_suffix	according to the grid

11.2.2 Detailed functionalities

Tag list

context : define the model using the xml file. Id is the only attribute accepted. Its value must be ”nemo” or ”n_nemo” for the nth AGRIF zoom. Child of simulation tag.

field : define the field to be output. Accepted attributes are axis_ref, description, enable, freq_op, grid_ref, id (if child of field_definition), level, operation, name, ref (if child of file), unit, zoom_ref. Child of field_definition, file or group of fields tag.

field_definition : definition of the part of the xml file corresponding to the field definition. Accept the same attributes as field tag. Child of context tag.

group : define a group of file or field. Accept the same attributes as file or field.

file : define the output fileÕs characteristics. Accepted attributes are description, enable, output_freq, output_level, id, name, name_suffix. Child of file_definition or group of files tag.

file_definition : definition of the part of the xml file corresponding to the file definition. Accept the same attributes as file tag. Child of context tag.

axis : definition of the vertical axis. Accepted attributes are description, id, positive, size, unit. Child of axis_definition tag.

axis_definition : definition of the part of the xml file corresponding to the vertical axis definition. Accept the same attributes as axis tag. Child of context tag

grid : definition of the horizontal grid. Accepted attributes are description and id. Child of axis_definition tag.

grid_definition : definition of the part of the xml file corresponding to the horizontal grid definition. Accept the same attributes as grid tag. Child of context tag

zoom : definition of a subdomain of an horizontal grid. Accepted attributes are description, id, i/jbegin, ni/j. Child of grid tag.

Attributes list

Applied to a tag or a group of tags.

Another table, perhaps ?

Attribute Applied to ? Definition Comment axis_ref field String defining the vertical axis of the variable. It refers to the id of the vertical axis defined in the axis tag. Use "non" if the variable has no vertical axis

axis_ref : field attribute. String defining the vertical axis of the variable. It refers to the id of the vertical axis defined in the axis tag. Use "none" if the variable has no vertical axis

description : this attribute can be applied to all tags but it is used only with the field tag. In this case, the value of description will be used to define, in the output netcdf file, the attributes long_name and standard_name of the variable.

enabled : field and file attribute. Logical to switch on/off the output of a field or a file.

freq_op : field attribute (automatically defined, see part 1.4 ("control of the xml attributes")). An integer defining the frequency in seconds at which NEMO is calling iom_put for this variable. It corresponds to the model time step (rn_rdt in the namelist) except for the variables computed at the frequency of the surface boundary condition (rn_rdt ? nn_fsbc in the namelist).

grid_ref : field attribute. String defining the horizontal grid of the variable. It refers to the id of the grid tag.

ibegin : zoom attribute. Integer defining the zoom starting point along x direction. Automatically defined for TAO/RAMA/PIRATA moorings (see part 1.4).

id : exists for all tag. This is a string defining the name to a specific tag that will be used later to refer to this tag. Tags of the same category must have different ids.

jbegin : zoom attribute. Integer defining the zoom starting point along y direction. Automatically defined for TAO/RAMA/PIRATA moorings and equatorial section (see part 1.4).

level : field attribute. Integer from 0 to 10 defining the output priority of a field. See output_level attribute definition

operation : field attribute. String defining the type of temporal operation to perform on a variable. Possible choices are "ave(X)" for temporal mean, "inst(X)" for instantaneous, "t_min(X)" for temporal min and "t_max(X)" for temporal max.

output_freq : file attribute. Integer defining the operation frequency in seconds. For example 86400 for daily mean.

output_level : file attribute. Integer from 0 to 10 defining the output priority of variables in a file : all variables listed in the file with a level smaller or equal to output_level will be output. Other variables wonÕt be output even if they are listed in the file.

positive : axis attribute (always .FALSE.). Logical defining the vertical axis convention used in *NEMO* (positive downward). Define the attribute positive of the variable in the netcdf output file.

prec : field attribute. Integer defining the output precision. Not implemented, we always output real4 arrays.

name : field or file attribute. String defining the name of a variable or a file. If the name of a file is undefined, its id is used as a name. 2 files must have different names. Files with specific ids will have their name automatically defined (see part 1.4). Note that is name will be automatically completed by the cpu number (if needed) and ".nc"

name_suffix : file attribute. String defining a suffix to be inserted after the name and before the cpu number and the ".nc" termination. Files with specific ids have an automatic definition of their suffix (see part 1.4).

ni : zoom attribute. Integer defining the zoom extent along x direction. Automatically defined for equatorial sections (see part 1.4).

nj : zoom attribute. Integer defining the zoom extent along x direction.

ref : field attribute. String referring to the id of the field we want to add in a file.

size : axis attribute. use unknown...

unit : field attribute. String defining the unit of a variable and the associated attribute in the netcdf output file.

zoom_ref : field attribute. String defining the subdomain of data on which the file should be written (to ouput data only in a limited area). It refers to the id of a zoom defined in the zoom tag.

11.2.3 IO_SERVER

Attached or detached mode ?

Iom_put is based on the io_server developed by Yann Meurdesoif from IPSL (see here for the source code or see its copy in NEMOGCM/EXTERNAL directory). This server can be used in "attached mode" (as a library) or in "detached mode" (as an external executable on n cpus). In attached mode, each cpu of NEMO will output its own subdomain. In detached mode, the io_server will gather data from NEMO and output them split over n files with n the number of cpu dedicated to the io_server.

Control the io_server : the namelist file xmlio_server.def

The control of the use of the io_server is done through the namelist file of the io_server called xmlio_server.def.

using_server : logical, switch to use the server in attached or detached mode (.TRUE. corresponding to detached mode).

using_oasis : logical, set to .TRUE. if NEMO is used in coupled mode.

client_id = "oceanx" : character, used only in coupled mode. Specify the id used in OASIS to refer to NEMO. The same id must be used to refer to NEMO in the $NBMODEL part of OASIS namcouple in the call of prim_init_comp_proto in cpl_oasis3f90

server_id = "ionemo" : character, used only in coupled mode. Specify the id used in OASIS to refer to the IO_SERVER when used in detached mode. Use the same id to refer to the io_server in the $NBMODEL part of OASIS namcouple.

global_mpi_buffer_size : integer ; define the size in Mb of the MPI buffer used by the io_server.

Number of cpu used by the io_server in detached mode

The number of cpu used by the io_server is specified only when launching the model. Here is an example of 2 cpus for the io_server and 6 cpu for opa using mpirun :

```
-p 2 -e ./ioserver
-p 6 -e ./opa
```

11.2.4 Practical issues

Add your own outputs

It is very easy to add you own outputs with iom_put. 4 points must be followed.

1- in NEMO code, add a
```
CALL iom_put( 'identifier', array )
```
where you want to output a 2D or 3D array.

2- don't forget to add
```
USE iom ! I/O manager library
```
in the list of used modules in the upper part of your module.

3- in the file_definition part of the xml file, add the definition of your variable using the same identifier you used in the f90 code.

```
<field_definition>
   ...
   <field id="identifier" description="blabla" />
   ...
</field_definition>
```

attributes axis_ref and grid_ref must be consistent with the size of the array to pass to iom_put. if your array is computed within the surface module each nn_fsbc time_step, add the field definition within the group defined with the id "SBC" : <group id="SBC"...>

4- add your field in one of the output files

```
<file id="file_1" .../>
   ...
   <field ref="identifier" />
   ...
</file>
```

Several time axes in the output file

If your output file contains variables with different operations (see operation definition), IOIPSL will create one specific time axis for each operation. Note that inst(X) will have a time axis corresponding to the end each output period whereas all other operators will have a time axis centred in the middle of the output periods.

Error/bug messages from IOIPSL

If you get the following error in the standard output file :

```
FATAL ERROR FROM ROUTINE flio_dom_set
 --> too many domains simultaneously defined
 --> please unset useless domains
 --> by calling flio_dom_unset
```

You must increase the value of dom_max_nb in fliocom.f90 (multiply it by 10 for example).

If you mix, in the same file, variables with different freq_op (see definition above), like for example variables from the surface module with other variables, IOIPSL will print in the standard output file warning messages saying there may be a bug.

```
WARNING FROM ROUTINE histvar_seq
 --> There were 10 errors in the learned sequence of variables
 --> for file   4
 --> This looks like a bug, please report it.
```

Don't worry, there is no bug, everything is properly working !

11.3 NetCDF4 Support (key_netcdf4)

Since version 3.3, support for NetCDF4 chunking and (loss-less) compression has been included. These options build on the standard NetCDF output and allow the user control over the size of the chunks via namelist settings. Chunking and compression can lead to significant reductions in file sizes for a small runtime overhead. For a fuller discussion on chunking and other performance issues the reader is referred to the NetCDF4 documentation found here.

The new features are only available when the code has been linked with a NetCDF4 library (version 4.1 onwards, recommended) which has been built with HDF5 support (version 1.8.4 onwards, recommended). Datasets created with chunking and compression are not backwards compatible with NetCDF3 "classic" format but most analysis codes can

be relinked simply with the new libraries and will then read both NetCDF3 and NetCDF4 files. NEMO executables linked with NetCDF4 libraries can be made to produce NetCDF3 files by setting the *ln_nc4zip* logical to false in the *namnc4* namelist :

```
!-----------------------------------------------------------------------
&namnc4        !   netcdf4 chunking and compression settings            ("key_netcdf4")
!-----------------------------------------------------------------------
   nn_nchunks_i=   4       !  number of chunks in i-dimension
   nn_nchunks_j=   4       !  number of chunks in j-dimension
   nn_nchunks_k=   31      !  number of chunks in k-dimension
                           !  setting nn_nchunks_k = jpk will give a chunk size of 1 in the vertical which
                           !  is optimal for postprocessing which works exclusively with horizontal slabs
   ln_nc4zip   = .true.    !  (T) use netcdf4 chunking and compression
                           !  (F) ignore chunking information and produce netcdf3-compatible files
/
```

If **key_netcdf4** has not been defined, these namelist parameters are not read. In this case, *ln_nc4zip* is set false and dummy routines for a few NetCDF4-specific functions are defined. These functions will not be used but need to be included so that compilation is possible with NetCDF3 libraries.

When using NetCDF4 libraries, **key_netcdf4** should be defined even if the intention is to create only NetCDF3-compatible files. This is necessary to avoid duplication between the dummy routines and the actual routines present in the library. Most compilers will fail at compile time when faced with such duplication. Thus when linking with NetCDF4 libraries the user must define **key_netcdf4** and control the type of NetCDF file produced via the namelist parameter.

Chunking and compression is applied only to 4D fields and there is no advantage in chunking across more than one time dimension since previously written chunks would have to be read back and decompressed before being added to. Therefore, user control over chunk sizes is provided only for the three space dimensions. The user sets an approximate number of chunks along each spatial axis. The actual size of the chunks will depend on global domain size for mono-processors or, more likely, the local processor domain size for distributed processing. The derived values are subject to practical minimum values (to avoid wastefully small chunk sizes) and cannot be greater than the domain size in any dimension. The algorithm used is :

```
ichunksz(1) = MIN( idomain_size,MAX( (idomain_size-1)/nn_nchunks_i + 1 ,16 ) )
ichunksz(2) = MIN( jdomain_size,MAX( (jdomain_size-1)/nn_nchunks_j + 1 ,16 ) )
ichunksz(3) = MIN( kdomain_size,MAX( (kdomain_size-1)/nn_nchunks_k + 1 , 1 ) )
ichunksz(4) = 1
```

As an example, setting :

```
nn_nchunks_i=4, nn_nchunks_j=4 and nn_nchunks_k=31
```

for a standard ORCA2_LIM configuration gives chunksizes of `46x38x1` respectively in the mono-processor case (i.e. global domain of `182x149x31`). An illustration of the potential space savings that NetCDF4 chunking and compression provides is given in table **??** which compares the results of two short runs of the ORCA2_LIM reference configuration with a 4x2 mpi partitioning. Note the variation in the compression ratio achieved which reflects chiefly the dry to wet volume ratio of each processing region.

Filename	NetCDF3 filesize (KB)	NetCDF4 filesize (KB)	Reduction %
ORCA2_restart_0000.nc	16420	8860	47%
ORCA2_restart_0001.nc	16064	11456	29%
ORCA2_restart_0002.nc	16064	9744	40%
ORCA2_restart_0003.nc	16420	9404	43%
ORCA2_restart_0004.nc	16200	5844	64%
ORCA2_restart_0005.nc	15848	8172	49%
ORCA2_restart_0006.nc	15848	8012	50%
ORCA2_restart_0007.nc	16200	5148	69%
ORCA2_2d_grid_T_0000.nc	2200	1504	32%
ORCA2_2d_grid_T_0001.nc	2200	1748	21%
ORCA2_2d_grid_T_0002.nc	2200	1592	28%
ORCA2_2d_grid_T_0003.nc	2200	1540	30%
ORCA2_2d_grid_T_0004.nc	2200	1204	46%
ORCA2_2d_grid_T_0005.nc	2200	1444	35%
ORCA2_2d_grid_T_0006.nc	2200	1428	36%
ORCA2_2d_grid_T_0007.nc	2200	1148	48%
...	...	...	..
ORCA2_2d_grid_W_0000.nc	4416	2240	50%
ORCA2_2d_grid_W_0001.nc	4416	2924	34%
ORCA2_2d_grid_W_0002.nc	4416	2512	44%
ORCA2_2d_grid_W_0003.nc	4416	2368	47%
ORCA2_2d_grid_W_0004.nc	4416	1432	68%
ORCA2_2d_grid_W_0005.nc	4416	1972	56%
ORCA2_2d_grid_W_0006.nc	4416	2028	55%
ORCA2_2d_grid_W_0007.nc	4416	1368	70%

TAB. 11.1 – Filesize comparison between NetCDF3 and NetCDF4 with chunking and compression

When **key_iomput** is activated with **key_netcdf4** chunking and compression parameters for fields produced via *iom_put* calls are set via an equivalent and identically named namelist to *namnc4* in *xmlio_server.def*. Typically this namelist serves the mean files whilst the *namnc4* in the main namelist file continues to serve the restart files. This duplication is unfortunate but appropriate since, if using io_servers, the domain sizes of the individual files produced by the io_server processes may be different to those produced by the invidual processing regions and different chunking choices may be desired.

11.4 Tracer/Dynamics Trends (key_trdtra, key_trddyn, key_trddvor, key_trdmld)

```
!-----------------------------------------------------------------------
&namtrd        !   diagnostics on dynamics and/or tracer trends         ("key_trddyn" and/or "key_trdtra")
!              !       or mixed-layer trends or barotropic vorticity    ("key_trdmld" or     "key_trdvor")
!-----------------------------------------------------------------------
   nn_trd      = 365       !  time step frequency dynamics and tracers trends
   nn_ctls     =   0       !  control surface type in mixed-layer trends (0,1 or n<jpk)
   rn_ucf      =   1.      !  unit conversion factor (=1 -> /seconds ; =86400. -> /day)
   cn_trdrst_in      = "restart_mld"   ! suffix of ocean restart name (input)
   cn_trdrst_out     = "restart_mld"   ! suffix of ocean restart name (output)
   ln_trdmld_restart = .false.         !  restart for ML diagnostics
   ln_trdmld_instant = .false.         !  flag to diagnose trends of instantantaneous or mean ML T/S
/
```

When **key_trddyn** and/or **key_trddyn** CPP variables are defined, each trend of the dynamics and/or temperature and salinity time evolution equations is stored in three-dimensional arrays just after their computation ($i.e.$ at the end of each $dyn \cdots .F90$ and/or $tra \cdots .F90$ routines). These trends are then used in *trdmod.F90* (see TRD directory) every *nn_trd* time-steps.

What is done depends on the CPP keys defined :

key_trddyn, key_trdtra : a check of the basin averaged properties of the momentum and/or tracer equations is performed ;

key_trdvor : a vertical summation of the moment tendencies is performed, then the curl is computed to obtain the barotropic vorticity tendencies which are output ;

key_trdmld : output of the tracer tendencies averaged vertically either over the mixed layer (*nn_ctls*=0), or over a fixed number of model levels (*nn_ctls*>1 provides the number of level), or over a spatially varying but temporally fixed number of levels (typically the base of the winter mixed layer) read in *ctlsurf_idx.nc* (*nn_ctls*=1) ;

The units in the output file can be changed using the *nn_ucf* namelist parameter. For example, in case of salinity tendency the units are given by PSU/s/*nn_ucf*. Setting *nn_ucf*=86400 ($i.e.$ the number of second in a day) provides the tendencies in PSU/d.

When **key_trdmld** is defined, two time averaging procedure are proposed. Setting *ln_trdmld_instant* to *true*, a simple time averaging is performed, so that the resulting tendency is the contribution to the change of a quantity between the two instantaneous values taken at the extremities of the time averaging period. Setting *ln_trdmld_instant* to *false*, a double time averaging is performed, so that the resulting tendency is the contribution to the change of a quantity between two *time mean* values. The later option requires the use of an extra file, *restart_mld.nc* (*ln_trdmld_restart*=true), to restart a run.

Note that the mixed layer tendency diagnostic can also be used on biogeochemical models via the **key_trdtrc** and **key_trdmld_trc** CPP keys.

11.5 On-line Floats trajectories (FLO) (key_floats)

```
!-----------------------------------------------------------------------
&namflo       !   float parameters                                    ("key_float")
!-----------------------------------------------------------------------
   ln_rstflo  = .false.    !  float restart (T) or not (F)
   nn_writefl =      75    !  frequency of writing in float output file
   nn_stockfl =    5475    !  frequency of creation of the float restart file
   ln_argo    = .false.    !  Argo type floats (stay at the surface each 10 days)
   ln_flork4  = .false.    !  trajectories computed with a 4th order Runge-Kutta (T)
                           !  or computed with Blanke' scheme (F)
/
```

The on-line computation of floats advected either by the three dimensional velocity field or constraint to remain at a given depth ($w = 0$ in the computation) have been introduced in the system during the CLIPPER project. The algorithm used is based either on the work of **?** (default option), or on a 4^th Runge-Hutta algorithm (*ln_flork4*=true). Note that the **?** algorithm have the advantage of providing trajectories which are consistent with the numeric of the code, so that the trajectories never intercept the bathymetry.

See also here the web site describing the off-line use of this marvellous diagnostic tool.

11.6 Transports across sections (key_diadct)

A module is available to compute volume, heat and salt tranports through sections. This diagnostic is actived with **key_diadct**.

Each section is defined by the coordinates of its 2 extremities. The pathway between them is contructed with the tools `SECTIONS_DIADCT` and is written in binary file `section_ijglobal.diadct_ORCA2_LIM` which might be read by NEMO to compute on-line transports.

On-line transports module will create three ascii files for outputs :

- `volume_transport` for volume transports (unit : $10^6 m^3 s^{-1}$)
- `heat_transport` for heat transports (unit : $10^{15} W$)
- `salt_transport` for salt transports (unit : $10^9 g s^{-1}$)

```
!-----------------------------------------------------------------------
&namdct        ! transports through sections
!-----------------------------------------------------------------------
    nn_dct      = 15       !  time step frequency for transports computing
    nn_dctwri   = 15       !  time step frequency for transports writing
    nn_secdebug =  0       !      0 : no section to debug
                           !     -1 : debug all section
                           !  0 < n : debug section number n
/
```

Concerning the on-line transports computing, some parameters are available in namelist :

- `nn_dct` is the frequency of instantaneous transports computing
- `nn_dctwri` is the frequency of writing (mean of instantaneous transports)
- `nn_debug` to debug a section

How to compute sections pathway and create the binary file ?

In NEMOGCM/TOOLS/SECTIONS_DIADCT/run, list_sections.ascii_global contains all the sections we want to compute ; this list of sections is inspired from MERSEA project metrics.

Another file is available for GYRE configuration (list_sections.ascii_GYRE).

Each section is defined by :

```
 long1 lat1 long2 lat2 nclass (ok/no)strpond (no)ice section_name
```

with :

- long1 lat1 , coordinates of the first extremity of the section ;
- long2 lat2 , coordinates of the second extremity of the section ;
- nclass the number of bounds of your classes (3 bounds for 2 classes, for example) ;
- okstrpond to compute heat and salt transport, nostrpond if no ;
- ice to compute surface and volume ice transports, noice if no.

Results of transports computing don't depend of the order of the 2 extremities in this file.

If nclass =/ 0,the next lines will containes the class type and the nclass bounds :

```
long1 lat1 long2 lat2 nclass (ok/no)strpond (no)ice section_name
classtype
zbound1
zbound2
.
.
nclass-1
nclass
```

where classtype can be :

- zsal for salinity classes
- ztem for temperature classes
- zlay for depth classes
- zsigi for insitu density classes
- zsigp for potential density classes

job.ksh compute the pathway for each section and create a binary file section_ijglobal.diadct_O which will be read by NEMO.

Example of two sections :

```
 -68. -54.5 -60. -64.7 00 okstrpond noice ACC_Drake_Passage
 -80.5 22.5 -80.5 25.5 05 nostrpond noice ATL_Cuba_Florida
ztem
-2.0
 4.5
 7.0
12.0
```

40.0

How to read the output files ?

The output format is :
date, time-step number, section number, section name, section slope coefficient, class number, class name, class bound 1 , classe bound2, transport_sens1 , transport_sens2, transport_total

If some sections have classes, the first lines correspond to the transport for each class and the last line corresponds to the total transport for all classes. If the other case, class number 1 corresponds to total class and this class is called N, like none.

transport_sens1 is the positive part of the transport (> = 0).
transport_sens2 is the negative part of the transport (< = 0).

section slope coefficient gives some informations about the significations of transports signs and sens :

section slope coefficient	section type	sens 1	sens 2	total transport
0.	horizontal	northward	southward	postive : northward
1000.	vertical	eastward	westward	postive : eastward
=/0, =/ 1000.	diagonal	eastward	westward	postive : eastward

11.7 Other Diagnostics (key_diahth, key_diaar5)

Aside from the standard model variables, other diagnostics can be computed on-line. The available ready-to-add diagnostics routines can be found in directory DIA. Among the available diagnostics the following ones are obtained when defining the **key_diahth** CPP key :

- the mixed layer depth (based on a density criterion, **?**) (*diahth.F90*)
- the turbocline depth (based on a turbulent mixing coefficient criterion) (*diahth.F90*)
- the depth of the 20˚C isotherm (*diahth.F90*)
- the depth of the thermocline (maximum of the vertical temperature gradient) (*diahth.F90*)

The poleward heat and salt transports, their advective and diffusive component, and the meriodional stream function can be computed on-line in *diaptr.F90* by setting *ln_diaptr* to true (see the *namptr* namelist below). When *ln_subbas* = true, transports and stream function are computed for the Atlantic, Indian, Pacific and Indo-Pacific Oceans (defined north of 30˚S) as well as for the World Ocean. The sub-basin decomposition requires an input file (*subbasins.nc*) which contains three 2D mask arrays, the Indo-Pacific mask been deduced from the sum of the Indian and Pacific mask (Fig **??**).

```
!-----------------------------------------------------------------------
&namptr       !   Poleward Transport Diagnostic
!-----------------------------------------------------------------------
```

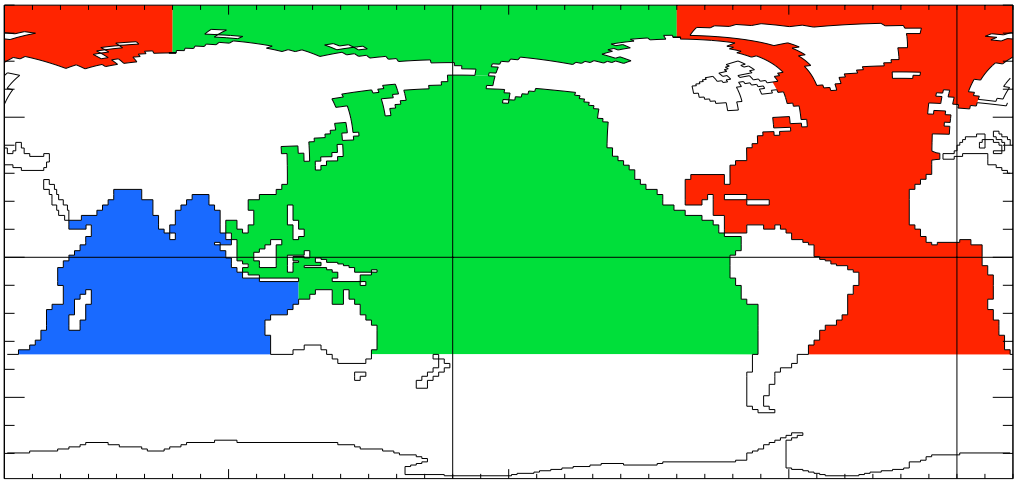

FIG. 11.1 – Decomposition of the World Ocean (here ORCA2) into sub-basin used in to compute the heat and salt transports as well as the meridional stream-function : Atlantic basin (red), Pacific basin (green), Indian basin (bleue), Indo-Pacific basin (bleue+green). Note that semi-enclosed seas (Red, Med and Baltic seas) as well as Hudson Bay are removed from the sub-basins. Note also that the Arctic Ocean has been split into Atlantic and Pacific basins along the North fold line.

```
   ln_diaptr  = .false.    !  Poleward heat and salt transport (T) or not (F)
   ln_diaznl  = .true.     !  Add zonal means and meridional stream functions
   ln_subbas  = .true.     !  Atlantic/Pacific/Indian basins computation (T) or not
                           !  (orca configuration only, need input basins mask file named "subbasins.nc"
   ln_ptrcomp = .true.     !  Add decomposition : overturning
   nn_fptr    =  1         !  Frequency of ptr computation [time step]
   nn_fwri    =  15        !  Frequency of ptr outputs [time step]
/
```

In addition, a series of diagnostics has been added in the *diaar5.F90*. They corresponds to outputs that are required for AR5 simulations (see Section **??** below for one of them). Activating those outputs requires to define the **key_diaar5** CPP key.

11.8 Diagnosing the Steric effect in sea surface height

Changes in steric sea level are caused when changes in the density of the water column imply an expansion or contraction of the column. It is essentially produced through surface heating/cooling and to a lesser extent through non-linear effects of the equation of state (cabbeling, thermobaricity...). Non-Boussinesq models contain all ocean effects within the ocean acting on the sea level. In particular, they include the steric effect. In

contrast, Boussinesq models, such as *NEMO* , conserve volume, rather than mass, and so do not properly represent expansion or contraction. The steric effect is therefore not explicitely represented. This approximation does not represent a serious error with respect to the flow field calculated by the model [**?**], but extra attention is required when investigating sea level, as steric changes are an important contribution to local changes in sea level on seasonal and climatic time scales. This is especially true for investigation into sea level rise due to global warming.

Fortunately, the steric contribution to the sea level consists of a spatially uniform component that can be diagnosed by considering the mass budget of the world ocean [**?**]. In order to better understand how global mean sea level evolves and thus how the steric sea level can be diagnosed, we compare, in the following, the non-Boussinesq and Boussinesq cases.

Let denote $\mathcal{M}$ the total mass of liquid seawater ($\mathcal{M} = \int_D \rho dv$), $\mathcal{V}$ the total volume of seawater ($\mathcal{V} = \int_D dv$), $\mathcal{A}$ the total surface of the ocean ($\mathcal{A} = \int_S ds$), $\bar{\rho}$ the global mean seawater (*in situ*) density ($\bar{\rho} = 1/\mathcal{V} \int_D \rho \, dv$), and $\bar{\eta}$ the global mean sea level ($\bar{\eta} = 1/\mathcal{A} \int_S \eta \, ds$).

A non-Boussinesq fluid conserves mass. It satisfies the following relations :

$$\begin{aligned} \mathcal{M} &= \mathcal{V} \, \bar{\rho} \\ \mathcal{V} &= \mathcal{A} \, \bar{\eta} \end{aligned} \tag{11.1}$$

Temporal changes in total mass is obtained from the density conservation equation :

$$\frac{1}{e_3} \partial_t (e_3 \, \rho) + \nabla(\rho \, \mathbf{U}) = \left. \frac{emp}{e_3} \right|_{surface} \tag{11.2}$$

where ρ is the *in situ* density, and *emp* the surface mass exchanges with the other media of the Earth system (atmosphere, sea-ice, land). Its global averaged leads to the total mass change

$$\partial_t \mathcal{M} = \mathcal{A} \, \overline{emp} \tag{11.3}$$

where $\overline{emp} = \int_S emp \, ds$ is the net mass flux through the ocean surface. Bringing (**??**) and the time derivative of (**??**) together leads to the evolution equation of the mean sea level

$$\partial_t \bar{\eta} = \frac{\overline{emp}}{\bar{\rho}} - \frac{\mathcal{V}}{\mathcal{A}} \, \frac{\partial_t \bar{\rho}}{\bar{\rho}} \tag{11.4}$$

The first term in equation (**??**) alters sea level by adding or subtracting mass from the ocean. The second term arises from temporal changes in the global mean density ; $i.e.$ from steric effects.

In a Boussinesq fluid, ρ is replaced by ρ_o in all the equation except when ρ appears multiplied by the gravity ($i.e.$ in the hydrostatic balance of the primitive Equations). In particular, the mass conservation equation, (**??**), degenerates into the incompressibility equation :

$$\frac{1}{e_3} \partial_t (e_3) + \nabla(\mathbf{U}) = \left. \frac{emp}{\rho_o \, e_3} \right|_{surface} \tag{11.5}$$

and the global average of this equation now gives the temporal change of the total volume,

$$\partial_t \mathcal{V} = \mathcal{A} \, \frac{\overline{emp}}{\rho_o} \tag{11.6}$$

Only the volume is conserved, not mass, or, more precisely, the mass which is conserved is the Boussinesq mass, $\mathcal{M}_o = \rho_o \mathcal{V}$. The total volume (or equivalently the global mean sea level) is altered only by net volume fluxes across the ocean surface, not by changes in mean mass of the ocean : the steric effect is missing in a Boussinesq fluid.

Nevertheless, following [**?**], the steric effect on the volume can be diagnosed by considering the mass budget of the ocean. The apparent changes in $\mathcal{M}$, mass of the ocean, which are not induced by surface mass flux must be compensated by a spatially uniform change in the mean sea level due to expansion/contraction of the ocean [**?**]. In others words, the Boussinesq mass, $\mathcal{M}_o$, can be related to $\mathcal{M}$, the total mass of the ocean seen by the Boussinesq model, via the steric contribution to the sea level, η_s, a spatially uniform variable, as follows :

$$\mathcal{M}_o = \mathcal{M} + \rho_o \, \eta_s \, \mathcal{A} \tag{11.7}$$

Any change in $\mathcal{M}$ which cannot be explained by the net mass flux through the ocean surface is converted into a mean change in sea level. Introducing the total density anomaly, $\mathcal{D} = \int_D d_a \, dv$, where $d_a = (\rho - \rho_o)/\rho_o$ is the density anomaly used in *NEMO* (cf. §5.8.1) in (**??**) leads to a very simple form for the steric height :

$$\eta_s = -\frac{1}{\mathcal{A}} \mathcal{D} \tag{11.8}$$

The above formulation of the steric height of a Boussinesq ocean requires four remarks. First, one can be tempted to define ρ_o as the initial value of $\mathcal{M}/\mathcal{V}$, $i.e.$ set $\mathcal{D}_{t=0} = 0$, so that the initial steric height is zero. We do not recommend that. Indeed, in this case ρ_o depends on the initial state of the ocean. Since ρ_o has a direct effect on the dynamics of the ocean (it appears in the pressure gradient term of the momentum equation) it is definitively not a good idea when inter-comparing experiments. We better recommend to fixe once for all ρ_o to $1035 \; Kg \, m^{-3}$. This value is a sensible choice for the reference density used in a Boussinesq ocean climate model since, with the exception of only a small percentage of the ocean, density in the World Ocean varies by no more than 2% from this value (**?**, page 47).

Second, we have assumed here that the total ocean surface, $\mathcal{A}$, does not change when the sea level is changing as it is the case in all global ocean GCMs (wetting and drying of grid point is not allowed).

Third, the discretisation of (**??**) depends on the type of free surface which is considered. In the non linear free surface case, $i.e.$ **key_vvl** defined, it is given by

$$\eta_s = -\frac{\sum_{i,\,j,\,k} d_a \; e_{1t} e_{2t} e_{3t}}{\sum_{i,\,j,\,k} e_{1t} e_{2t} e_{3t}} \tag{11.9}$$

whereas in the linear free surface, the volume above the *z=0* surface must be explicitly taken into account to better approximate the total ocean mass and thus the steric sea level :

$$\eta_s = -\frac{\sum_{i,\,j,\,k} d_a \; e_{1t} e_{2t} e_{3t} + \sum_{i,\,j} d_a \; e_{1t} e_{2t} \eta}{\sum_{i,\,j,\,k} e_{1t} e_{2t} e_{3t} + \sum_{i,\,j} e_{1t} e_{2t} \eta} \tag{11.10}$$

The fourth and last remark concerns the effective sea level and the presence of sea-ice. In the real ocean, sea ice (and snow above it) depresses the liquid seawater through its mass loading. This depression is a result of the mass of sea ice/snow system acting on the liquid ocean. There is, however, no dynamical effect associated with these depressions in the liquid ocean sea level, so that there are no associated ocean currents. Hence, the dynamically relevant sea level is the effective sea level, $i.e.$ the sea level as if sea ice (and snow) were converted to liquid seawater [**?**]. However, in the current version of *NEMO* the sea-ice is levitating above the ocean without mass exchanges between ice and ocean. Therefore the model effective sea level is always given by $\eta + \eta_s$, whether or not there is sea ice present.

In AR5 outputs, the thermosteric sea level is demanded. It is steric sea level due to changes in ocean density arising just from changes in temperature. It is given by :

$$\eta_s = -\frac{1}{\mathcal{A}} \int_D d_a(T, S_o, p_o) \, dv \tag{11.11}$$

where S_o and p_o are the initial salinity and pressure, respectively.

Both steric and thermosteric sea level are computed in *diaar5.F90* which needs the **key_diaar5** defined to be called.

12 Observation and model comparison (OBS)

Authors : D. Lea, M. Martin, K. Mogensen, A. Vidard, A. Weaver...

Contents

The observation and model comparison code (OBS) reads in observation files (profile temperature and salinity, sea surface temperature, sea level anomaly, sea ice concentration, and velocity) and calculates an interpolated model equivalent value at the observation location and nearest model timestep. The OBS code is called from *opa.F90* in order to initialise the model and to calculate the model equivalent values for observations on the 0th timestep. The code is then called again after each timestep from *step.F90*. The code was originally developed for use with NEMOVAR.

For all data types a 2D horizontal interpolator is needed to interpolate the model fields to the observation location. For *in situ* profiles, a 1D vertical interpolator is needed in addition to provide model fields at the observation depths. Currently this only works in z-level model configurations, but is being developed to work with a generalised vertical coordinate system. Temperature data from moored buoys (TAO, TRITON, PIRATA) in the ENACT/ENSEMBLES data-base are available as daily averaged quantities. For this type of observation the observation operator will compare such observations to the model temperature fields averaged over one day. The relevant observation type may be specified in the namelist using *endailyavtypes*. Otherwise the model value from the nearest timestep to the observation time is used.

The resulting data are saved in a "feedback" file (or files) which can be used for model validation and verification and also to provide information for data assimilation. This code is controlled by the namelist *nam_obs*. To build with the OBS code active **key_diaobs** must be set.

Section 11.1 introduces a test example of the observation operator code including where to obtain data and how to setup the namelist. Section 11.2 introduces some more technical details of the different observation types used and also shows a more complete namelist. Finally section 11.3 introduces some of the theoretical aspects of the observation operator including interpolation methods and running on multiple processors.

12.1 Running the observation operator code example

This section describes an example of running the observation operator code using profile data which can be freely downloaded. It shows how to adapt an existing run and build of NEMO to run the observation operator.

1. Compile NEMO with **key_diaobs** set.
2. Download some ENSEMBLES EN3 data from http ://www.hadobs.org. Choose observations which are valid for the period of your test run because the observation operator compares the model and observations for a matching date and time.
3. Add the following to the NEMO namelist to run the observation operator on this data. Set the *enactfiles* namelist parameter to the observation file name (or link in to *profiles_01ṅc*) :

```
!-----------------------------------------------------------------------
!       namobs    observation usage switch
!-----------------------------------------------------------------------
!
!  ln_t3d                  Logical switch for T profile observations
!  ln_s3d                  Logical switch for S profile observations
!  ln_ena                  Logical switch for ENACT insitu data set
!  ln_cor                  Logical switch for Coriolis insitu data set
!  ln_profb                Logical switch for feedback insitu data set
!  ln_sla                  Logical switch for SLA observations
!  ln_sladt                Logical switch for AVISO SLA data
!  ln_slafb                Logical switch for feedback SLA data
!  ln_ssh                  Logical switch for SSH observations
!  ln_sst                  Logical switch for SST observations
!  ln_reysst               Logical switch for Reynolds observations
!  ln_ghrsst               Logical switch for GHRSST observations
!  ln_sstfb                Logical switch for feedback SST data
!  ln_sss                  Logical switch for SSS observations
!  ln_seaice               Logical switch for Sea Ice observations
!  ln_vel3d                Logical switch for velocity observations
!  ln_velavcur             Logical switch for velocity daily av. cur.
!  ln_velhrcur             Logical switch for velocity high freq. cur.
!  ln_velavadcp            Logical switch for velocity daily av. ADCP
!  ln_velhradcp            Logical switch for velocity high freq. ADCP
!  ln_velfb                Logical switch for feedback velocity data
!  ln_grid_global          Global distribtion of observations
!  ln_grid_search_lookup   Logical switch for obs grid search w/lookup table
!  grid_search_file        Grid search lookup file header
!  enactfiles              ENACT input observation file names
!  coriofiles              Coriolis input observation file name
!  profbfiles              Profile feedback input observation file name
!  ln_profb_enatim         Enact feedback input time setting switch
!  slafilesact             Active SLA input observation file name
!  slafilespas             Passive SLA input observation file name
!  slafbfiles              Feedback SLA input observation file name
!  sstfiles                GHRSST input observation file name
!  sstfbfiles              Feedback SST input observation file name
!  seaicefiles             Sea Ice input observation file name
!  velavcurfiles           Vel. cur. daily av. input file name
!  velhvcurfiles           Vel. cur. high freq. input file name
!  velavadcpfiles          Vel. ADCP daily av. input file name
!  velhvadcpfiles          Vel. ADCP high freq. input file name
!  velfbfiles              Vel. feedback input observation file name
!  dobsini                 Initial date in window YYYYMMDD.HHMMSS
!  dobsend                 Final date in window YYYYMMDD.HHMMSS
!  n1dint                  Type of vertical interpolation method
!  n2dint                  Type of horizontal interpolation method
!  ln_nea                  Rejection of observations near land switch
!  nmsshc                  MSSH correction scheme
!  mdtcorr                 MDT  correction
!  mdtcutoff               MDT cutoff for computed correction
!  ln_altbias              Logical switch for alt bias
!  ln_ignmis               Logical switch for ignoring missing files
!  endailyavtypes          ENACT daily average types
 &namobs
   ln_t3d = .true.
   ln_s3d = .true.
   ln_ena = .true.
   enactfiles = 'profiles_01.nc'
   ln_grid_global = .true.
   ln_grid_search_lookup = .true.
   ln_ignmis = .true.
/
```

The option *ln_t3d* and *ln_s3d* switch on the temperature and salinity profile observation operator code. The *ln_ena* switch turns on the reading of ENACT/ENSEMBLES type profile data. The filename or array of filenames are specified using the *enactfiles* variable. The model grid points for a particular observation latitude and longitude are found using the grid searching part of the code. This can be expensive, particularly for large numbers of observations, setting *ln_grid_search_lookup* allows the use of a lookup table which is saved into an "xypos" file (or files). This will need to be generated the first time if it does not exist in the run directory. However, once produced it will significantly speed up future grid searches. Setting *ln_grid_global* means that the code distributes the observations evenly between processors. Alternatively each processor will work with observations located within the model subdomain.

The NEMOVAR system contains utilities to plot the feedback files, convert and recombine the files. These are available on request from the NEMOVAR team.

12.2 Technical details

Here we show a more complete example namelist and also show the NetCDF headers of the observation files that may be used with the observation operator

```
!-----------------------------------------------------------------------
&namobs       !  observation usage switch                              ('key_diaobs')
!-----------------------------------------------------------------------
   ln_t3d     = .false.    ! Logical switch for T profile observations
   ln_s3d     = .false.    ! Logical switch for S profile observations
   ln_ena     = .false.    ! Logical switch for ENACT insitu data set
   !                       !     ln_cor                Logical switch for Coriolis insitu data set
   ln_profb   = .false.    ! Logical switch for feedback insitu data set
   ln_sla     = .false.    ! Logical switch for SLA observations

   ln_sladt   = .false.    ! Logical switch for AVISO SLA data

   ln_slafb   = .false.    ! Logical switch for feedback SLA data
                           !     ln_ssh                Logical switch for SSH observations

   ln_sst     = .false.    ! Logical switch for SST observations
                           !     ln_reysst             Logical switch for Reynolds observations
                           !     ln_ghrsst             Logical switch for GHRSST observations

   ln_sstfb   = .false.    ! Logical switch for feedback SST data
                           !     ln_sss                Logical switch for SSS observations
                           !     ln_seaice             Logical switch for Sea Ice observations
                           !     ln_vel3d              Logical switch for velocity observations
                           !     ln_velavcur           Logical switch for velocity daily av. cur.
                           !     ln_velhrcur           Logical switch for velocity high freq. cur.
                           !     ln_velavadcp          Logical switch for velocity daily av. ADCP
                           !     ln_velhradcp          Logical switch for velocity high freq. ADCP
                           !     ln_velfb              Logical switch for feedback velocity data
                           !     ln_grid_global        Global distribtion of observations
                           !     ln_grid_search_lookup Logical switch for obs grid search w/lookup table
                           !     grid_search_file      Grid search lookup file header
                           !     enactfiles            ENACT input observation file names
                           !     coriofiles            Coriolis input observation file name
   !                       ! profbfiles: Profile feedback input observation file name
   profbfiles = 'profiles_01.nc'
                           !     ln_profb_enatim       Enact feedback input time setting switch
                           !     slafilesact           Active SLA input observation file name
                           !     slafilespas           Passive SLA input observation file name
   !                       ! slafbfiles: Feedback SLA input observation file name
   slafbfiles = 'sla_01.nc'
                           !     sstfiles              GHRSST input observation file name
   !                       ! sstfbfiles: Feedback SST input observation file name
   sstfbfiles = 'sst_01.nc' 'sst_02.nc' 'sst_03.nc' 'sst_04.nc' 'sst_05.nc'
                           !     seaicefiles           Sea Ice input observation file name
                           !     velavcurfiles         Vel. cur. daily av. input file name
                           !     velhvcurfiles         Vel. cur. high freq. input file name
                           !     velavadcpfiles        Vel. ADCP daily av. input file name
                           !     velhvadcpfiles        Vel. ADCP high freq. input file name
                           !     velfbfiles            Vel. feedback input observation file name
                           !     dobsini               Initial date in window YYYYMMDD.HHMMSS
                           !     dobsend               Final date in window YYYYMMDD.HHMMSS
                           !     n1dint                Type of vertical interpolation method
                           !     n2dint                Type of horizontal interpolation method
                           !     ln_nea                Rejection of observations near land switch
   nmsshc     = 0          ! MSSH correction scheme
                           !     mdtcorr               MDT  correction
                           !     mdtcutoff             MDT cutoff for computed correction
   ln_altbias = .false.    ! Logical switch for alt bias
   ln_ignmis  = .true.     ! Logical switch for ignoring missing files
                           !     endailyavtypes   ENACT daily average types
   ln_grid_global = .true.
   ln_grid_search_lookup = .false.
/
```

This name list uses the "feedback" type observation file input format for profile, sea level anomaly and sea surface temperature data. All the observation files must be in NetCDF

format. Some example headers (produced using *ncdump -h*) for profile data, sea level anomaly and sea surface temperature are in the following subsections.

12.2.1 Profile feedback type observation file header

```
netcdf profiles_01 {
dimensions:
     N_OBS = 603 ;
     N_LEVELS = 150 ;
     N_VARS = 2 ;
     N_QCF = 2 ;
     N_ENTRIES = 1 ;
     N_EXTRA = 1 ;
     STRINGNAM = 8 ;
     STRINGGRID = 1 ;
     STRINGWMO = 8 ;
     STRINGTYP = 4 ;
     STRINGJULD = 14 ;
variables:
     char VARIABLES(N_VARS, STRINGNAM) ;
          VARIABLES:long_name = "List of variables in feedback files" ;
     char ENTRIES(N_ENTRIES, STRINGNAM) ;
          ENTRIES:long_name = "List of additional entries for each variable in feedback files" ;
     char EXTRA(N_EXTRA, STRINGNAM) ;
          EXTRA:long_name = "List of extra variables" ;
     char STATION_IDENTIFIER(N_OBS, STRINGWMO) ;
          STATION_IDENTIFIER:long_name = "Station identifier" ;
     char STATION_TYPE(N_OBS, STRINGTYP) ;
          STATION_TYPE:long_name = "Code instrument type" ;
     double LONGITUDE(N_OBS) ;
          LONGITUDE:long_name = "Longitude" ;
          LONGITUDE:units = "degrees_east" ;
          LONGITUDE:_Fillvalue = 99999.f ;
     double LATITUDE(N_OBS) ;
          LATITUDE:long_name = "Latitude" ;
          LATITUDE:units = "degrees_north" ;
          LATITUDE:_Fillvalue = 99999.f ;
     double DEPTH(N_OBS, N_LEVELS) ;
          DEPTH:long_name = "Depth" ;
          DEPTH:units = "metre" ;
          DEPTH:_Fillvalue = 99999.f ;
     int DEPTH_QC(N_OBS, N_LEVELS) ;
          DEPTH_QC:long_name = "Quality on depth" ;
          DEPTH_QC:Conventions = "q where q =[0,9]" ;
          DEPTH_QC:_Fillvalue = 0 ;
     int DEPTH_QC_FLAGS(N_OBS, N_LEVELS, N_QCF) ;
          DEPTH_QC_FLAGS:long_name = "Quality flags on depth" ;
          DEPTH_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
     double JULD(N_OBS) ;
          JULD:long_name = "Julian day" ;
          JULD:units = "days since JULD_REFERENCE" ;
          JULD:Conventions = "relative julian days with decimal part (as parts of day)" ;
          JULD:_Fillvalue = 99999.f ;
     char JULD_REFERENCE(STRINGJULD) ;
          JULD_REFERENCE:long_name = "Date of reference for julian days" ;
          JULD_REFERENCE:Conventions = "YYYYMMDDHHMMSS" ;
     int OBSERVATION_QC(N_OBS) ;
          OBSERVATION_QC:long_name = "Quality on observation" ;
          OBSERVATION_QC:Conventions = "q where q =[0,9]" ;
          OBSERVATION_QC:_Fillvalue = 0 ;
     int OBSERVATION_QC_FLAGS(N_OBS, N_QCF) ;
          OBSERVATION_QC_FLAGS:long_name = "Quality flags on observation" ;
          OBSERVATION_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
          OBSERVATION_QC_FLAGS:_Fillvalue = 0 ;
     int POSITION_QC(N_OBS) ;
          POSITION_QC:long_name = "Quality on position (latitude and longitude)" ;
          POSITION_QC:Conventions = "q where q =[0,9]" ;
          POSITION_QC:_Fillvalue = 0 ;
     int POSITION_QC_FLAGS(N_OBS, N_QCF) ;
          POSITION_QC_FLAGS:long_name = "Quality flags on position" ;
          POSITION_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
          POSITION_QC_FLAGS:_Fillvalue = 0 ;
     int JULD_QC(N_OBS) ;
          JULD_QC:long_name = "Quality on date and time" ;
          JULD_QC:Conventions = "q where q =[0,9]" ;
          JULD_QC:_Fillvalue = 0 ;
     int JULD_QC_FLAGS(N_OBS, N_QCF) ;
```

```
          JULD_QC_FLAGS:long_name = "Quality flags on date and time" ;
          JULD_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
          JULD_QC_FLAGS:_Fillvalue = 0 ;
     int ORIGINAL_FILE_INDEX(N_OBS) ;
          ORIGINAL_FILE_INDEX:long_name = "Index in original data file" ;
          ORIGINAL_FILE_INDEX:_Fillvalue = -99999 ;
     float POTM_OBS(N_OBS, N_LEVELS) ;
          POTM_OBS:long_name = "Potential temperature" ;
          POTM_OBS:units = "Degrees Celsius" ;
          POTM_OBS:_Fillvalue = 99999.f ;
     float POTM_Hx(N_OBS, N_LEVELS) ;
          POTM_Hx:long_name = "Model interpolated potential temperature" ;
          POTM_Hx:units = "Degrees Celsius" ;
          POTM_Hx:_Fillvalue = 99999.f ;
     int POTM_QC(N_OBS) ;
          POTM_QC:long_name = "Quality on potential temperature" ;
          POTM_QC:Conventions = "q where q =[0,9]" ;
          POTM_QC:_Fillvalue = 0 ;
     int POTM_QC_FLAGS(N_OBS, N_QCF) ;
          POTM_QC_FLAGS:long_name = "Quality flags on potential temperature" ;
          POTM_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
          POTM_QC_FLAGS:_Fillvalue = 0 ;
     int POTM_LEVEL_QC(N_OBS, N_LEVELS) ;
          POTM_LEVEL_QC:long_name = "Quality for each level on potential temperature" ;
          POTM_LEVEL_QC:Conventions = "q where q =[0,9]" ;
          POTM_LEVEL_QC:_Fillvalue = 0 ;
     int POTM_LEVEL_QC_FLAGS(N_OBS, N_LEVELS, N_QCF) ;
          POTM_LEVEL_QC_FLAGS:long_name = "Quality flags for each level on potential temperature" ;
          POTM_LEVEL_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
          POTM_LEVEL_QC_FLAGS:_Fillvalue = 0 ;
     int POTM_IOBSI(N_OBS) ;
          POTM_IOBSI:long_name = "ORCA grid search I coordinate" ;
     int POTM_IOBSJ(N_OBS) ;
          POTM_IOBSJ:long_name = "ORCA grid search J coordinate" ;
     int POTM_IOBSK(N_OBS, N_LEVELS) ;
          POTM_IOBSK:long_name = "ORCA grid search K coordinate" ;
     char POTM_GRID(STRINGGRID) ;
          POTM_GRID:long_name = "ORCA grid search grid (T,U,V)" ;
     float PSAL_OBS(N_OBS, N_LEVELS) ;
          PSAL_OBS:long_name = "Practical salinity" ;
          PSAL_OBS:units = "PSU" ;
          PSAL_OBS:_Fillvalue = 99999.f ;
     float PSAL_Hx(N_OBS, N_LEVELS) ;
          PSAL_Hx:long_name = "Model interpolated practical salinity" ;
          PSAL_Hx:units = "PSU" ;
          PSAL_Hx:_Fillvalue = 99999.f ;
     int PSAL_QC(N_OBS) ;
          PSAL_QC:long_name = "Quality on practical salinity" ;
          PSAL_QC:Conventions = "q where q =[0,9]" ;
          PSAL_QC:_Fillvalue = 0 ;
     int PSAL_QC_FLAGS(N_OBS, N_QCF) ;
          PSAL_QC_FLAGS:long_name = "Quality flags on practical salinity" ;
          PSAL_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
          PSAL_QC_FLAGS:_Fillvalue = 0 ;
     int PSAL_LEVEL_QC(N_OBS, N_LEVELS) ;
          PSAL_LEVEL_QC:long_name = "Quality for each level on practical salinity" ;
          PSAL_LEVEL_QC:Conventions = "q where q =[0,9]" ;
          PSAL_LEVEL_QC:_Fillvalue = 0 ;
     int PSAL_LEVEL_QC_FLAGS(N_OBS, N_LEVELS, N_QCF) ;
          PSAL_LEVEL_QC_FLAGS:long_name = "Quality flags for each level on practical salinity" ;
          PSAL_LEVEL_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
          PSAL_LEVEL_QC_FLAGS:_Fillvalue = 0 ;
     int PSAL_IOBSI(N_OBS) ;
          PSAL_IOBSI:long_name = "ORCA grid search I coordinate" ;
     int PSAL_IOBSJ(N_OBS) ;
          PSAL_IOBSJ:long_name = "ORCA grid search J coordinate" ;
     int PSAL_IOBSK(N_OBS, N_LEVELS) ;
          PSAL_IOBSK:long_name = "ORCA grid search K coordinate" ;
     char PSAL_GRID(STRINGGRID) ;
          PSAL_GRID:long_name = "ORCA grid search grid (T,U,V)" ;
     float TEMP(N_OBS, N_LEVELS) ;
          TEMP:long_name = "Insitu temperature" ;
          TEMP:units = "Degrees Celsius" ;
          TEMP:_Fillvalue = 99999.f ;

// global attributes:
          :title = "NEMO observation operator output" ;
          :Convention = "NEMO unified observation operator output" ;
}
```

12.2.2 Sea level anomaly feedback type observation file header

```
netcdf sla_01 {
dimensions:
     N_OBS = 41301 ;
     N_LEVELS = 1 ;
     N_VARS = 1 ;
     N_QCF = 2 ;
     N_ENTRIES = 1 ;
     N_EXTRA = 1 ;
     STRINGNAM = 8 ;
     STRINGGRID = 1 ;
     STRINGWMO = 8 ;
     STRINGTYP = 4 ;
     STRINGJULD = 14 ;
variables:
     char VARIABLES(N_VARS, STRINGNAM) ;
          VARIABLES:long_name = "List of variables in feedback files" ;
     char ENTRIES(N_ENTRIES, STRINGNAM) ;
          ENTRIES:long_name = "List of additional entries for each variable in feedback files" ;
     char EXTRA(N_EXTRA, STRINGNAM) ;
          EXTRA:long_name = "List of extra variables" ;
     char STATION_IDENTIFIER(N_OBS, STRINGWMO) ;
          STATION_IDENTIFIER:long_name = "Station identifier" ;
     char STATION_TYPE(N_OBS, STRINGTYP) ;
          STATION_TYPE:long_name = "Code instrument type" ;
     double LONGITUDE(N_OBS) ;
          LONGITUDE:long_name = "Longitude" ;
          LONGITUDE:units = "degrees_east" ;
          LONGITUDE:_Fillvalue = 99999.f ;
     double LATITUDE(N_OBS) ;
          LATITUDE:long_name = "Latitude" ;
          LATITUDE:units = "degrees_north" ;
          LATITUDE:_Fillvalue = 99999.f ;
     double DEPTH(N_OBS, N_LEVELS) ;
          DEPTH:long_name = "Depth" ;
          DEPTH:units = "metre" ;
          DEPTH:_Fillvalue = 99999.f ;
     int DEPTH_QC(N_OBS, N_LEVELS) ;
          DEPTH_QC:long_name = "Quality on depth" ;
          DEPTH_QC:Conventions = "q where q =[0,9]" ;
          DEPTH_QC:_Fillvalue = 0 ;
     int DEPTH_QC_FLAGS(N_OBS, N_LEVELS, N_QCF) ;
          DEPTH_QC_FLAGS:long_name = "Quality flags on depth" ;
          DEPTH_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
     double JULD(N_OBS) ;
          JULD:long_name = "Julian day" ;
          JULD:units = "days since JULD_REFERENCE" ;
          JULD:Conventions = "relative julian days with decimal part (as parts of day)" ;
          JULD:_Fillvalue = 99999.f ;
     char JULD_REFERENCE(STRINGJULD) ;
          JULD_REFERENCE:long_name = "Date of reference for julian days" ;
          JULD_REFERENCE:Conventions = "YYYYMMDDHHMMSS" ;
     int OBSERVATION_QC(N_OBS) ;
          OBSERVATION_QC:long_name = "Quality on observation" ;
          OBSERVATION_QC:Conventions = "q where q =[0,9]" ;
          OBSERVATION_QC:_Fillvalue = 0 ;
     int OBSERVATION_QC_FLAGS(N_OBS, N_QCF) ;
          OBSERVATION_QC_FLAGS:long_name = "Quality flags on observation" ;
          OBSERVATION_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
          OBSERVATION_QC_FLAGS:_Fillvalue = 0 ;
     int POSITION_QC(N_OBS) ;
          POSITION_QC:long_name = "Quality on position (latitude and longitude)" ;
          POSITION_QC:Conventions = "q where q =[0,9]" ;
          POSITION_QC:_Fillvalue = 0 ;
     int POSITION_QC_FLAGS(N_OBS, N_QCF) ;
          POSITION_QC_FLAGS:long_name = "Quality flags on position" ;
          POSITION_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
          POSITION_QC_FLAGS:_Fillvalue = 0 ;
     int JULD_QC(N_OBS) ;
          JULD_QC:long_name = "Quality on date and time" ;
          JULD_QC:Conventions = "q where q =[0,9]" ;
          JULD_QC:_Fillvalue = 0 ;
     int JULD_QC_FLAGS(N_OBS, N_QCF) ;
          JULD_QC_FLAGS:long_name = "Quality flags on date and time" ;
          JULD_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
          JULD_QC_FLAGS:_Fillvalue = 0 ;
     int ORIGINAL_FILE_INDEX(N_OBS) ;
          ORIGINAL_FILE_INDEX:long_name = "Index in original data file" ;
          ORIGINAL_FILE_INDEX:_Fillvalue = -99999 ;
     float SLA_OBS(N_OBS, N_LEVELS) ;
```

```
          SLA_OBS:long_name = "Sea level anomaly" ;
          SLA_OBS:units = "metre" ;
          SLA_OBS:_Fillvalue = 99999.f ;
     float SLA_Hx(N_OBS, N_LEVELS) ;
          SLA_Hx:long_name = "Model interpolated sea level anomaly" ;
          SLA_Hx:units = "metre" ;
          SLA_Hx:_Fillvalue = 99999.f ;
     int SLA_QC(N_OBS) ;
          SLA_QC:long_name = "Quality on sea level anomaly" ;
          SLA_QC:Conventions = "q where q =[0,9]" ;
          SLA_QC:_Fillvalue = 0 ;
     int SLA_QC_FLAGS(N_OBS, N_QCF) ;
          SLA_QC_FLAGS:long_name = "Quality flags on sea level anomaly" ;
          SLA_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
          SLA_QC_FLAGS:_Fillvalue = 0 ;
     int SLA_LEVEL_QC(N_OBS, N_LEVELS) ;
          SLA_LEVEL_QC:long_name = "Quality for each level on sea level anomaly" ;
          SLA_LEVEL_QC:Conventions = "q where q =[0,9]" ;
          SLA_LEVEL_QC:_Fillvalue = 0 ;
     int SLA_LEVEL_QC_FLAGS(N_OBS, N_LEVELS, N_QCF) ;
          SLA_LEVEL_QC_FLAGS:long_name = "Quality flags for each level on sea level anomaly" ;
          SLA_LEVEL_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
          SLA_LEVEL_QC_FLAGS:_Fillvalue = 0 ;
     int SLA_IOBSI(N_OBS) ;
          SLA_IOBSI:long_name = "ORCA grid search I coordinate" ;
     int SLA_IOBSJ(N_OBS) ;
          SLA_IOBSJ:long_name = "ORCA grid search J coordinate" ;
     int SLA_IOBSK(N_OBS, N_LEVELS) ;
          SLA_IOBSK:long_name = "ORCA grid search K coordinate" ;
     char SLA_GRID(STRINGGRID) ;
          SLA_GRID:long_name = "ORCA grid search grid (T,U,V)" ;
     float MDT(N_OBS, N_LEVELS) ;
          MDT:long_name = "Mean Dynamic Topography" ;
          MDT:units = "metre" ;
          MDT:_Fillvalue = 99999.f ;

// global attributes:
          :title = "NEMO observation operator output" ;
          :Convention = "NEMO unified observation operator output" ;
}
```

The mean dynamic topography (MDT) must be provided in a separate file defined on the model grid called *slaReferenceLevel.nc*. The MDT is required in order to produce the model equivalent sea level anomaly from the model sea surface height. Below is an example header for this file (on the ORCA025 grid).

```
dimensions:
        x = 1442 ;
        y = 1021 ;
variables:
        float nav_lon(y, x) ;
                nav_lon:units = "degrees_east" ;
        float nav_lat(y, x) ;
                nav_lat:units = "degrees_north" ;
        float sossheig(y, x) ;
                sossheig:_FillValue = -1.e+30f ;
                sossheig:coordinates = "nav_lon nav_lat" ;
                sossheig:long_name = "Mean Dynamic Topography" ;
                sossheig:units = "metres" ;
                sossheig:grid = "orca025T" ;
```

12.2.3 Sea surface temperature feedback type observation file header

```
netcdf sst_01 {
dimensions:
     N_OBS = 33099 ;
     N_LEVELS = 1 ;
     N_VARS = 1 ;
     N_QCF = 2 ;
     N_ENTRIES = 1 ;
```

```
     STRINGNAM = 8 ;
     STRINGGRID = 1 ;
     STRINGWMO = 8 ;
     STRINGTYP = 4 ;
     STRINGJULD = 14 ;
variables:
     char VARIABLES(N_VARS, STRINGNAM) ;
          VARIABLES:long_name = "List of variables in feedback files" ;
     char ENTRIES(N_ENTRIES, STRINGNAM) ;
          ENTRIES:long_name = "List of additional entries for each variable in feedback files" ;
     char STATION_IDENTIFIER(N_OBS, STRINGWMO) ;
          STATION_IDENTIFIER:long_name = "Station identifier" ;
     char STATION_TYPE(N_OBS, STRINGTYP) ;
          STATION_TYPE:long_name = "Code instrument type" ;
     double LONGITUDE(N_OBS) ;
          LONGITUDE:long_name = "Longitude" ;
          LONGITUDE:units = "degrees_east" ;
          LONGITUDE:_Fillvalue = 99999.f ;
     double LATITUDE(N_OBS) ;
          LATITUDE:long_name = "Latitude" ;
          LATITUDE:units = "degrees_north" ;
          LATITUDE:_Fillvalue = 99999.f ;
     double DEPTH(N_OBS, N_LEVELS) ;
          DEPTH:long_name = "Depth" ;
          DEPTH:units = "metre" ;
          DEPTH:_Fillvalue = 99999.f ;
     int DEPTH_QC(N_OBS, N_LEVELS) ;
          DEPTH_QC:long_name = "Quality on depth" ;
          DEPTH_QC:Conventions = "q where q =[0,9]" ;
          DEPTH_QC:_Fillvalue = 0 ;
     int DEPTH_QC_FLAGS(N_OBS, N_LEVELS, N_QCF) ;
          DEPTH_QC_FLAGS:long_name = "Quality flags on depth" ;
          DEPTH_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
     double JULD(N_OBS) ;
          JULD:long_name = "Julian day" ;
          JULD:units = "days since JULD_REFERENCE" ;
          JULD:Conventions = "relative julian days with decimal part (as parts of day)" ;
          JULD:_Fillvalue = 99999.f ;
     char JULD_REFERENCE(STRINGJULD) ;
          JULD_REFERENCE:long_name = "Date of reference for julian days" ;
          JULD_REFERENCE:Conventions = "YYYYMMDDHHMMSS" ;
     int OBSERVATION_QC(N_OBS) ;
          OBSERVATION_QC:long_name = "Quality on observation" ;
          OBSERVATION_QC:Conventions = "q where q =[0,9]" ;
          OBSERVATION_QC:_Fillvalue = 0 ;
     int OBSERVATION_QC_FLAGS(N_OBS, N_QCF) ;
          OBSERVATION_QC_FLAGS:long_name = "Quality flags on observation" ;
          OBSERVATION_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
          OBSERVATION_QC_FLAGS:_Fillvalue = 0 ;
     int POSITION_QC(N_OBS) ;
          POSITION_QC:long_name = "Quality on position (latitude and longitude)" ;
          POSITION_QC:Conventions = "q where q =[0,9]" ;
          POSITION_QC:_Fillvalue = 0 ;
     int POSITION_QC_FLAGS(N_OBS, N_QCF) ;
          POSITION_QC_FLAGS:long_name = "Quality flags on position" ;
          POSITION_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
          POSITION_QC_FLAGS:_Fillvalue = 0 ;
     int JULD_QC(N_OBS) ;
          JULD_QC:long_name = "Quality on date and time" ;
          JULD_QC:Conventions = "q where q =[0,9]" ;
          JULD_QC:_Fillvalue = 0 ;
     int JULD_QC_FLAGS(N_OBS, N_QCF) ;
          JULD_QC_FLAGS:long_name = "Quality flags on date and time" ;
          JULD_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
          JULD_QC_FLAGS:_Fillvalue = 0 ;
     int ORIGINAL_FILE_INDEX(N_OBS) ;
          ORIGINAL_FILE_INDEX:long_name = "Index in original data file" ;
          ORIGINAL_FILE_INDEX:_Fillvalue = -99999 ;
     float SST_OBS(N_OBS, N_LEVELS) ;
          SST_OBS:long_name = "Sea surface temperature" ;
          SST_OBS:units = "Degree centigrade" ;
          SST_OBS:_Fillvalue = 99999.f ;
     float SST_Hx(N_OBS, N_LEVELS) ;
          SST_Hx:long_name = "Model interpolated sea surface temperature" ;
          SST_Hx:units = "Degree centigrade" ;
          SST_Hx:_Fillvalue = 99999.f ;
     int SST_QC(N_OBS) ;
          SST_QC:long_name = "Quality on sea surface temperature" ;
          SST_QC:Conventions = "q where q =[0,9]" ;
          SST_QC:_Fillvalue = 0 ;
     int SST_QC_FLAGS(N_OBS, N_QCF) ;
          SST_QC_FLAGS:long_name = "Quality flags on sea surface temperature" ;
          SST_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
```

```
          SST_QC_FLAGS:_Fillvalue = 0 ;
     int SST_LEVEL_QC(N_OBS, N_LEVELS) ;
          SST_LEVEL_QC:long_name = "Quality for each level on sea surface temperature" ;
          SST_LEVEL_QC:Conventions = "q where q =[0,9]" ;
          SST_LEVEL_QC:_Fillvalue = 0 ;
     int SST_LEVEL_QC_FLAGS(N_OBS, N_LEVELS, N_QCF) ;
          SST_LEVEL_QC_FLAGS:long_name = "Quality flags for each level on sea surface temperature" ;
          SST_LEVEL_QC_FLAGS:Conventions = "NEMOVAR flag conventions" ;
          SST_LEVEL_QC_FLAGS:_Fillvalue = 0 ;
     int SST_IOBSI(N_OBS) ;
          SST_IOBSI:long_name = "ORCA grid search I coordinate" ;
     int SST_IOBSJ(N_OBS) ;
          SST_IOBSJ:long_name = "ORCA grid search J coordinate" ;
     int SST_IOBSK(N_OBS, N_LEVELS) ;
          SST_IOBSK:long_name = "ORCA grid search K coordinate" ;
     char SST_GRID(STRINGGRID) ;
          SST_GRID:long_name = "ORCA grid search grid (T,U,V)" ;

// global attributes:
          :title = "NEMO observation operator output" ;
          :Convention = "NEMO unified observation operator output" ;
}
```

12.3 Theoretical details

12.3.1 Horizontal interpolation methods

Consider an observation point P with with longitude and latitude $(\lambda_{\mathrm{P}}, \phi_{\mathrm{P}})$ and the four nearest neighbouring model grid points A, B, C and D with longitude and latitude (λ_{A}, ϕ_{A}), (λ_{B}, ϕ_{B}) etc. All horizontal interpolation methods implemented in NEMO estimate the value of a model variable x at point P as a weighted linear combination of the values of the model variables at the grid points A, B etc. :

$$x_{\mathrm{P}} = \frac{1}{w}\left(w_{\mathrm{A}} x_{\mathrm{A}} + w_{\mathrm{B}} x_{\mathrm{B}} + w_{\mathrm{C}} x_{\mathrm{C}} + w_{\mathrm{D}} x_{\mathrm{D}}\right) \tag{12.1}$$

where w_{A}, w_{B} etc. are the respective weights for the model field at points A, B etc., and $w = w_{\mathrm{A}} + w_{\mathrm{B}} + w_{\mathrm{C}} + w_{\mathrm{D}}$.

Four different possibilities are available for computing the weights.

1. **Great-Circle distance-weighted interpolation.** The weights are computed as a function of the great-circle distance $s(P, \cdot)$ between P and the model grid points A, B etc. For example, the weight given to the field x_{A} is specified as the product of the distances from P to the other points :

   $$w_{\mathrm{A}} = s(\mathrm{P}, \mathrm{B})\, s(\mathrm{P}, \mathrm{C})\, s(\mathrm{P}, \mathrm{D})$$

   where

   $$s\,(\mathrm{P}, \mathrm{M}) = \cos^{-1}\{\sin\phi_{\mathrm{P}} \sin\phi_{\mathrm{M}} + \cos\phi_{\mathrm{P}} \cos\phi_{\mathrm{M}} \cos(\lambda_{\mathrm{M}} - \lambda_{\mathrm{P}})\} \tag{12.2}$$

   and M corresponds to B, C or D. A more stable form of the great-circle distance formula for small distances (x near 1) involves the arcsine function ($e.g.$ see p. 101 of **?** :

   $$s\,(\mathrm{P}, \mathrm{M}) = \sin^{-1}\left\{\sqrt{1 - x^2}\right\}$$

where

$$x = a_{\mathrm{M}} a_{\mathrm{P}} + b_{\mathrm{M}} b_{\mathrm{P}} + c_{\mathrm{M}} c_{\mathrm{P}}$$

and

$$\begin{aligned} a_{\mathrm{M}} &= \sin\phi_{\mathrm{M}}, \\ a_{\mathrm{P}} &= \sin\phi_{\mathrm{P}}, \\ b_{\mathrm{M}} &= \cos\phi_{\mathrm{M}} \cos\phi_{\mathrm{M}}, \\ b_{\mathrm{P}} &= \cos\phi_{\mathrm{P}} \cos\phi_{\mathrm{P}}, \\ c_{\mathrm{M}} &= \cos\phi_{\mathrm{M}} \sin\phi_{\mathrm{M}}, \\ c_{\mathrm{P}} &= \cos\phi_{\mathrm{P}} \sin\phi_{\mathrm{P}}. \end{aligned}$$

2. **Great-Circle distance-weighted interpolation with small angle approximation.** Similar to the previous interpolation but with the distance s computed as

$$s\left(\mathrm{P},\mathrm{M}\right) = \sqrt{\left(\phi_{\mathrm{M}} - \phi_{\mathrm{P}}\right)^2 + \left(\lambda_{\mathrm{M}} - \lambda_{\mathrm{P}}\right)^2 \cos^2\phi_{\mathrm{M}}} \tag{12.3}$$

   where M corresponds to A, B, C or D.

3. **Bilinear interpolation for a regular spaced grid.** The interpolation is split into two 1D interpolations in the longitude and latitude directions, respectively.

4. **Bilinear remapping interpolation for a general grid.** An iterative scheme that involves first mapping a quadrilateral cell into a cell with coordinates (0,0), (1,0), (0,1) and (1,1). This method is based on the SCRIP interpolation package [**?**].

12.3.2 Grid search

For many grids used by the NEMO model, such as the ORCA family, the horizontal grid coordinates i and j are not simple functions of latitude and longitude. Therefore, it is not always straightforward to determine the grid points surrounding any given observational position. Before the interpolation can be performed, a search algorithm is then required to determine the corner points of the quadrilateral cell in which the observation is located. This is the most difficult and time consuming part of the 2D interpolation procedure. A robust test for determining if an observation falls within a given quadrilateral cell is as follows. Let $\mathrm{P}(\lambda_{\mathrm{P}}, \phi_{\mathrm{P}})$ denote the observation point, and let $\mathrm{A}(\lambda_{\mathrm{A}}, \phi_{\mathrm{A}})$, $\mathrm{B}(\lambda_{\mathrm{B}}, \phi_{\mathrm{B}})$, $\mathrm{C}(\lambda_{\mathrm{C}}, \phi_{\mathrm{C}})$ and $\mathrm{D}(\lambda_{\mathrm{D}}, \phi_{\mathrm{D}})$ denote the bottom left, bottom right, top left and top right corner points of the cell, respectively. To determine if P is inside the cell, we verify that the cross-products

$$\begin{aligned} \mathbf{r}_{\mathrm{PA}} \times \mathbf{r}_{\mathrm{PC}} &= \left[(\lambda_{\mathrm{A}} - \lambda_{\mathrm{P}})(\phi_{\mathrm{C}} - \phi_{\mathrm{P}}) - (\lambda_{\mathrm{C}} - \lambda_{\mathrm{P}})(\phi_{\mathrm{A}} - \phi_{\mathrm{P}})\right] \widehat{\mathbf{k}} \\ \mathbf{r}_{\mathrm{PB}} \times \mathbf{r}_{\mathrm{PA}} &= \left[(\lambda_{\mathrm{B}} - \lambda_{\mathrm{P}})(\phi_{\mathrm{A}} - \phi_{\mathrm{P}}) - (\lambda_{\mathrm{A}} - \lambda_{\mathrm{P}})(\phi_{\mathrm{B}} - \phi_{\mathrm{P}})\right] \widehat{\mathbf{k}} \\ \mathbf{r}_{\mathrm{PC}} \times \mathbf{r}_{\mathrm{PD}} &= \left[(\lambda_{\mathrm{C}} - \lambda_{\mathrm{P}})(\phi_{\mathrm{D}} - \phi_{\mathrm{P}}) - (\lambda_{\mathrm{D}} - \lambda_{\mathrm{P}})(\phi_{\mathrm{C}} - \phi_{\mathrm{P}})\right] \widehat{\mathbf{k}} \\ \mathbf{r}_{\mathrm{PD}} \times \mathbf{r}_{\mathrm{PB}} &= \left[(\lambda_{\mathrm{D}} - \lambda_{\mathrm{P}})(\phi_{\mathrm{B}} - \phi_{\mathrm{P}}) - (\lambda_{\mathrm{B}} - \lambda_{\mathrm{P}})(\phi_{\mathrm{D}} - \phi_{\mathrm{P}})\right] \widehat{\mathbf{k}} \end{aligned} \tag{12.4}$$

point in the opposite direction to the unit normal $\widehat{\mathbf{k}}$ (i.e., that the coefficients of $\widehat{\mathbf{k}}$ are negative), where $\mathbf{r}_{\mathrm{PA}}$, $\mathbf{r}_{\mathrm{PB}}$, etc. correspond to the vectors between points P and A, P and B, etc.. The method used is similar to the method used in the SCRIP interpolation package [**?**].

In order to speed up the grid search, there is the possibility to construct a lookup table for a user specified resolution. This lookup table contains the lower and upper bounds on the i and j indices to be searched for on a regular grid. For each observation position, the closest point on the regular grid of this position is computed and the i and j ranges of this point searched to determine the precise four points surrounding the observation.

12.3.3 Parallel aspects of horizontal interpolation

For horizontal interpolation, there is the basic problem that the observations are unevenly distributed on the globe. In numerical models, it is common to divide the model grid into subgrids (or domains) where each subgrid is executed on a single processing element with explicit message passing for exchange of information along the domain boundaries when running on a massively parallel processor (MPP) system. This approach is used by *NEMO* .

For observations there is no natural distribution since the observations are not equally distributed on the globe. Two options have been made available : 1) geographical distribution ; and 2) round-robin.

Geographical distribution of observations among processors

This is the simplest option in which the observations are distributed according to the domain of the grid-point parallelization. Figure 11.1 shows an example of the distribution of the *in situ* data on processors with a different colour for each observation on a given processor for a 4×2 decomposition with ORCA2. The grid-point domain decomposition is clearly visible on the plot.

The advantage of this approach is that all information needed for horizontal interpolation is available without any MPP communication. Of course, this is under the assumption that we are only using a 2×2 grid-point stencil for the interpolation (e.g., bilinear interpolation). For higher order interpolation schemes this is no longer valid. A disadvantage with the above scheme is that the number of observations on each processor can be very different. If the cost of the actual interpolation is expensive relative to the communication of data needed for interpolation, this could lead to load imbalance.

Round-robin distribution of observations among processors

An alternative approach is to distribute the observations equally among processors and use message passing in order to retrieve the stencil for interpolation. The simplest distribution of the observations is to distribute them using a round-robin scheme. Figure 11.2 shows the distribution of the *in situ* data on processors for the round-robin distribution of observations with a different colour for each observation on a given processor for a 4×2

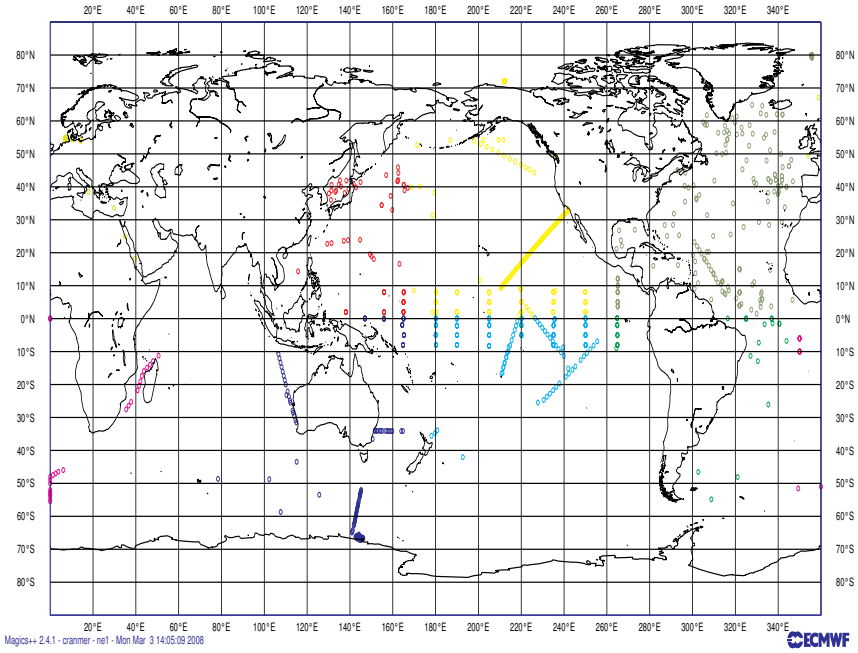

FIG. 12.1 – Example of the distribution of observations with the geographical distribution of observational data.

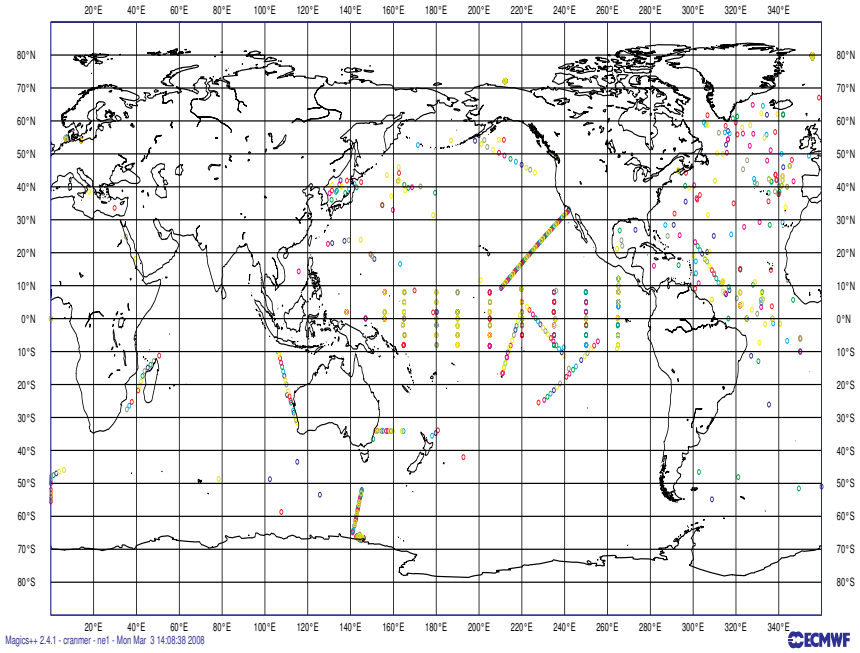

FIG. 12.2 – Example of the distribution of observations with the round-robin distribution of observational data.

decomposition with ORCA2 for the same input data as in Fig. 11.1. The observations are now clearly randomly distributed on the globe. In order to be able to perform horizontal interpolation in this case, a subroutine has been developed that retrieves any grid points in the global space.

12.3.4 Vertical interpolation operator

The vertical interpolation is achieved using either a cubic spline or linear interpolation. For the cubic spline, the top and bottom boundary conditions for the second derivative of the interpolating polynomial in the spline are set to zero. At the bottom boundary, this is done using the land-ocean mask.

13 Apply assimilation increments (ASM)

Authors : D. Lea, M. Martin, K. Mogensen, A. Weaver, ...

Contents

The ASM code adds the functionality to apply increments to the model variables : temperature, salinity, sea surface height, velocity and sea ice concentration. These are read into the model from a NetCDF file which may be produced by data assimilation. The code can also output model background fields which are used as an input to data assimilation code. This is all controlled by the namelist *nam_asminc*. There is a brief description of all the namelist options provided. To build the ASM code **key_asminc** must be set.

13.1 Direct initialization

Direct initialization (DI) refers to the instantaneous correction of the model background state using the analysis increment. DI is used when *ln_asmdin* is set to true.

13.2 Incremental Analysis Updates

Rather than updating the model state directly with the analysis increment, it may be preferable to introduce the increment gradually into the ocean model in order to minimize spurious adjustment processes. This technique is referred to as Incremental Analysis Updates (IAU) [**?**]. IAU is a common technique used with 3D assimilation methods such as 3D-Var or OI. IAU is used when *ln_asmiau* is set to true.

With IAU, the model state trajectory $\mathbf{x}$ in the assimilation window ($t_0 \leq t_i \leq t_N$) is corrected by adding the analysis increments for temperature, salinity, horizontal velocity and SSH as additional tendency terms to the prognostic equations :

$$\mathbf{x}^a(t_i) = M(t_i, t_0)[\mathbf{x}^b(t_0)] \; + \; F_i \delta \tilde{\mathbf{x}}^a \tag{13.1}$$

where F_i is a weighting function for applying the increments $\delta \tilde{\mathbf{x}}^a$ defined such that $\sum_{i=1}^{N} F_i = 1$. $\mathbf{x}^b$ denotes the model initial state and $\mathbf{x}^a$ is the model state after the increments are applied. To control the adjustment time of the model to the increment, the increment can be applied over an arbitrary sub-window, $t_m \leq t_i \leq t_n$, of the main assimilation window, where $t_0 \leq t_m \leq t_i$ and $t_i \leq t_n \leq t_N$, Typically the increments are spread evenly over the full window. In addition, two different weighting functions have been implemented. The first function employs constant weights,

$$F_i^{(1)} = \begin{cases} 0 & \text{if} \quad t_i < t_m \\ 1/M & \text{if} \quad t_m < t_i \leq t_n \\ 0 & \text{if} \quad t_i > t_n \end{cases} \tag{13.2}$$

where $M = m - n$. The second function employs peaked hat-like weights in order to give maximum weight in the centre of the sub-window, with the weighting reduced linearly to

a small value at the window end-points :

$$F_i^{(2)} = \begin{cases} 0 & \text{if} \quad t_i < t_m \\ \alpha\, i & \text{if} \quad t_m \leq t_i \leq t_{M/2} \\ \alpha\,(M - i + 1) & \text{if} \quad t_{M/2} < t_i \leq t_n \\ 0 & \text{if} \quad t_i > t_n \end{cases} \tag{13.3}$$

where $\alpha^{-1} = \sum_{i=1}^{M/2} 2i$ and M is assumed to be even. The weights described by (12.3) provide a smoother transition of the analysis trajectory from one assimilation cycle to the next than that described by (12.2).

13.3 Implementation details

Here we show an example namelist and the header of an example assimilation increments file on the ORCA2 grid.

```
!-----------------------------------------------------------------------
&nam_asminc   !   assimilation increments                          ('key_asminc')
!-----------------------------------------------------------------------
    ln_bkgwri = .false.    !  Logical switch for writing out background state
    ln_trjwri = .false.    !  Logical switch for writing out state trajectory
    ln_trainc = .false.    !  Logical switch for applying tracer increments
    ln_dyninc = .false.    !  Logical switch for applying velocity increments
    ln_sshinc = .false.    !  Logical switch for applying SSH increments
    ln_asmdin = .false.    !  Logical switch for Direct Initialization (DI)
    ln_asmiau = .false.    !  Logical switch for Incremental Analysis Updating (IAU)
    nitbkg    = 0          !  Timestep of background in [0,nitend-nit000-1]
    nitdin    = 0          !  Timestep of background for DI in [0,nitend-nit000-1]
    nitiaustr = 1          !  Timestep of start of IAU interval in [0,nitend-nit000-1]
    nitiaufin = 15         !  Timestep of end of IAU interval in [0,nitend-nit000-1]
    niaufn    = 0          !  Type of IAU weighting function
    nittrjfrq = 0          !  Frequency of trajectory output for 4D-VAR
    ln_salfix = .false.    !  Logical switch for ensuring that the sa > salfixmin
    salfixmin = -9999      !  Minimum salinity after applying the increments
/
```

The header of an assimilation increments file produced using the NetCDF tool *ncdump -h* is shown below

```
netcdf assim_background_increments {
dimensions:
        x = 182 ;
        y = 149 ;
        z = 31 ;
        t = UNLIMITED ; // (1 currently)
variables:
        float nav_lon(y, x) ;
        float nav_lat(y, x) ;
        float nav_lev(z) ;
        double time_counter(t) ;
        double time ;
        double z_inc_dateb ;
        double z_inc_datef ;
        double bckint(t, z, y, x) ;
        double bckins(t, z, y, x) ;
        double bckinu(t, z, y, x) ;
        double bckinv(t, z, y, x) ;
        double bckineta(t, y, x) ;

// global attributes:
                :DOMAIN_number_total = 1 ;
                :DOMAIN_number = 0 ;
                :DOMAIN_dimensions_ids = 1, 2 ;
                :DOMAIN_size_global = 182, 149 ;
```

```
                :DOMAIN_size_local = 182, 149 ;
                :DOMAIN_position_first = 1, 1 ;
                :DOMAIN_position_last = 182, 149 ;
                :DOMAIN_halo_size_start = 0, 0 ;
                :DOMAIN_halo_size_end = 0, 0 ;
                :DOMAIN_type = "BOX" ;
}
```

14 Miscellaneous Topics

Contents

14.1 Representation of Unresolved Straits

In climate modeling, it often occurs that a crucial connections between water masses is broken as the grid mesh is too coarse to resolve narrow straits. For example, coarse grid spacing typically closes off the Mediterranean from the Atlantic at the Strait of Gibraltar. In this case, it is important for climate models to include the effects of salty water entering the Atlantic from the Mediterranean. Likewise, it is important for the Mediterranean to replenish its supply of water from the Atlantic to balance the net evaporation occurring over the Mediterranean region. This problem occurs even in eddy permitting simulations. For example, in ORCA 1/4˚several straits of the Indonesian archipelago (Ombai, Lombok...) are much narrow than even a single ocean grid-point.

We describe briefly here the three methods that can be used in *NEMO* to handle such improperly resolved straits. The first two consist of opening the strait by hand while ensuring that the mass exchanges through the strait are not too large by either artificially reducing the surface of the strait grid-cells or, locally increasing the lateral friction. In the third one, the strait is closed but exchanges of mass, heat and salt across the land are allowed. Note that such modifications are so specific to a given configuration that no attempt has been made to set them in a generic way. However, examples of how they can be set up is given in the ORCA 2˚and 0.5˚configurations (search for **key_orca_r2** or **key_orca_r05** in the code).

14.1.1 Hand made geometry changes

• reduced scale factor in the cross-strait direction to a value in better agreement with the true mean width of the strait. (Fig. 13.1). This technique is sometime called ”partially open face” or ”partially closed cells”. The key issue here is only to reduce the faces of T-cell ($i.e.$ change the value of the horizontal scale factors at u- or v-point) but not the volume of the T-cell. Indeed, reducing the volume of strait T-cell can easily produce a numerical instability at that grid point that would require a reduction of the model time step. The changes associated with strait management are done in *domhgr.F90*, just after the definition or reading of the horizontal scale factors.

• increase of the viscous boundary layer thickness by local increase of the fmask value at the coast (Fig. 13.1). This is done in *dommsk.F90* together with the setting of the coastal value of fmask (see Section 8.1)

14.1.2 Cross Land Advection (*tracla.F90*)

```
!-----------------------------------------------------------------------
&namcla        !   cross land advection
!-----------------------------------------------------------------------
   nn_cla      =    0      !  advection between 2 ocean pts separates by land
/
```

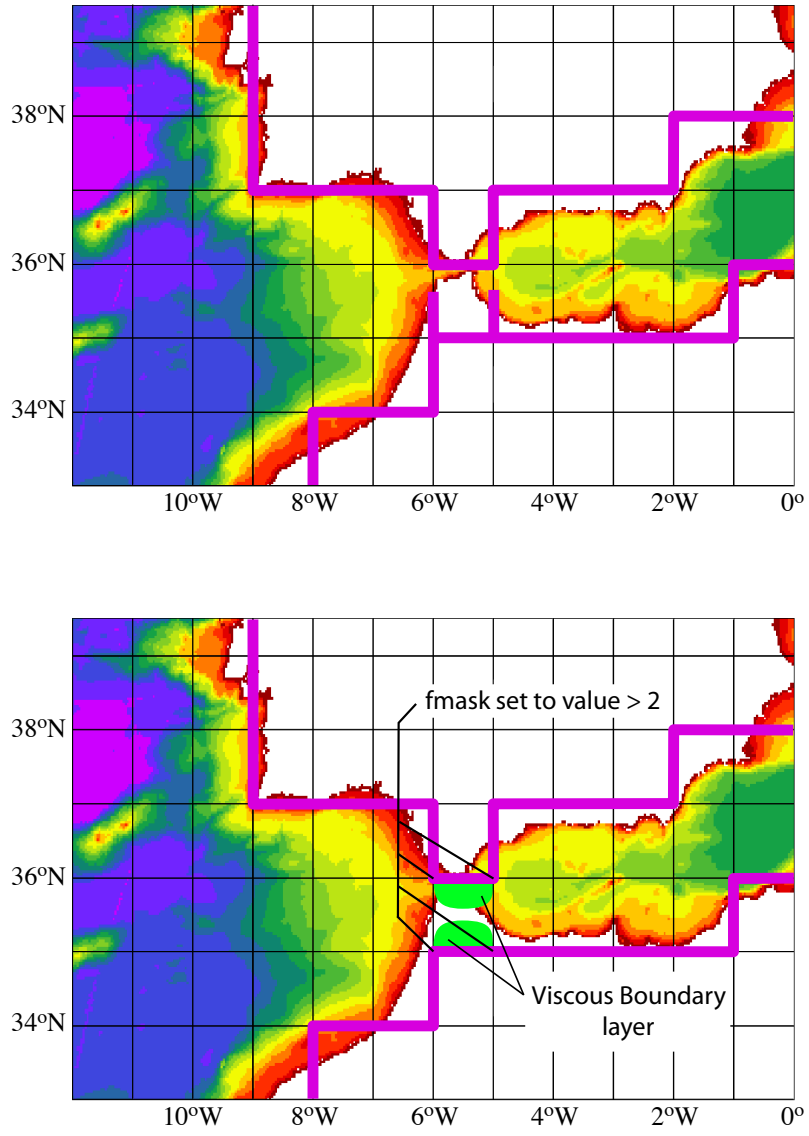

FIG. 14.1 – Example of the Gibraltar strait defined in a $1^\circ \times 1^\circ$ mesh. *Top* : using partially open cells. The meridional scale factor at v-point is reduced on both sides of the strait to account for the real width of the strait (about 20 km). Note that the scale factors of the strait T-point remains unchanged. *Bottom* : using viscous boundary layers. The four fmask parameters along the strait coastlines are set to a value larger than 4, *i.e.* "strong" no-slip case (see Fig.8.2) creating a large viscous boundary layer that allows a reduced transport through the strait.

Add a short description of CLA staff here or in lateral boundary condition chapter ?

14.2 Closed seas (*closea.F90*)

Add here a short description of the way closed seas are managed

14.3 Sub-Domain Functionality (*jpizoom*, *jpjzoom*)

The sub-domain functionality, also improperly called the zoom option (improperly because it is not associated with a change in model resolution) is a quite simple function that allows a simulation over a sub-domain of an already defined configuration ($i.e.$ without defining a new mesh, initial state and forcings). This option can be useful for testing the user settings of surface boundary conditions, or the initial ocean state of a huge ocean model configuration while having a small computer memory requirement. It can also be used to easily test specific physics in a sub-domain (for example, see [**?**] for a test of the coupling used in the global ocean version of OPA between sea-ice and ocean model over the Arctic or Antarctic ocean, using a sub-domain). In the standard model, this option does not include any specific treatment for the ocean boundaries of the sub-domain : they are considered as artificial vertical walls. Nevertheless, it is quite easy to add a restoring term toward a climatology in the vicinity of such boundaries (see §5.6).

In order to easily define a sub-domain over which the computation can be performed, the dimension of all input arrays (ocean mesh, bathymetry, forcing, initial state, ...) are defined as *jpidta*, *jpjdta* and *jpkdta* (*par_oce.F90* module), while the computational domain is defined through *jpiglo*, *jpjglo* and *jpk* (*par_oce.F90* module). When running the model over the whole domain, the user sets *jpiglo=jpidta jpjglo=jpjdta* and *jpk=jpkdta*. When running the model over a sub-domain, the user has to provide the size of the sub-domain, (*jpiglo*, *jpjglo*, *jpkglo*), and the indices of the south western corner as *jpizoom* and *jpjzoom* in the *par_oce.F90* module (Fig. 13.2).

Note that a third set of dimensions exist, *jpi*, *jpj* and *jpk* which is actually used to perform the computation. It is set by default to *jpi=jpjglo* and *jpj=jpjglo*, except for massively parallel computing where the computational domain is laid out on local processor memories following a 2D horizontal splitting.

14.4 Accelerating the Convergence (*nn_acc* = 1)

```
!-----------------------------------------------------------------------
&namdom        !   space and time domain (bathymetry, mesh, timestep)
!-----------------------------------------------------------------------
   nn_bathy    =    1      !  compute (=0) or read (=1) the bathymetry file
   nn_closea    =   0      !  remove (=0) or keep (=1) closed seas and lakes (ORCA)
   nn_msh      =    0      !  create (=1) a mesh file or not (=0)
   rn_hmin     =   -3.     !  min depth of the ocean (>0) or min number of ocean level (<0)
   rn_e3zps_min=   20.     !  partial step thickness is set larger than the minimum of
   rn_e3zps_rat=    0.1    !  rn_e3zps_min and rn_e3zps_rat*e3t, with 0<rn_e3zps_rat<1
                           !
   rn_rdt      = 5760.     !  time step for the dynamics (and tracer if nn_acc=0)
   nn_baro     =   64      !  number of barotropic time step           ("key_dynspg_ts")
```

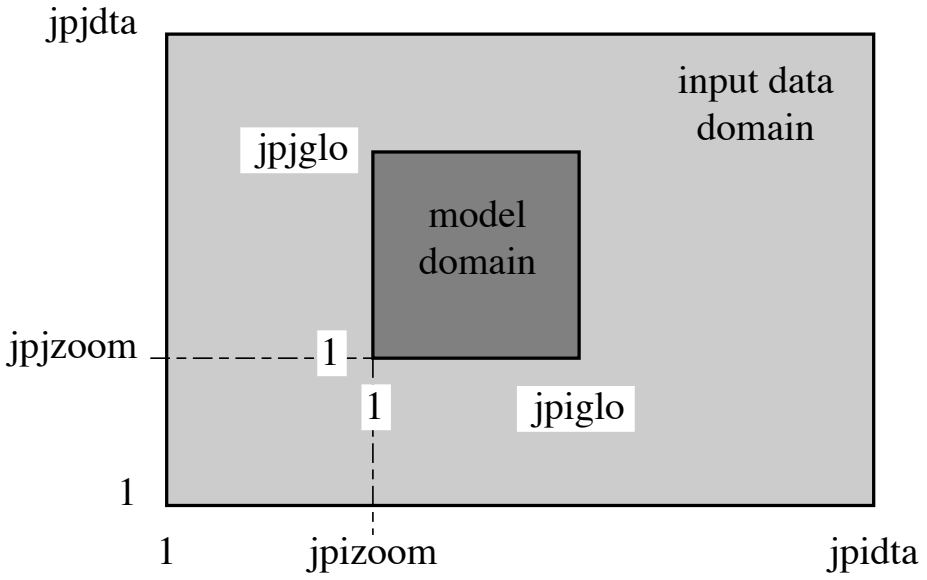

FIG. 14.2 – Position of a model domain compared to the data input domain when the zoom functionality is used.

```
   rn_atfp     =    0.1    !  asselin time filter parameter
   nn_acc      =    0      !  acceleration of convergence : =1      used, rdt < rdttra(k)
                              !                              =0, not used, rdt = rdttra
   rn_rdtmin   = 28800.         !  minimum time step on tracers (used if nn_acc=1)
   rn_rdtmax   = 28800.         !  maximum time step on tracers (used if nn_acc=1)
   rn_rdth     =  800.          !  depth variation of tracer time step  (used if nn_acc=1)
/
```

Searching an equilibrium state with an global ocean model requires a very long time integration period (a few thousand years for a global model). Due to the size of the time step required for numerical stability (less than a few hours), this usually requires a large elapsed time. In order to overcome this problem, **?** introduces a technique that is intended to accelerate the spin up to equilibrium. It uses a larger time step in the tracer evolution equations than in the momentum evolution equations. It does not affect the equilibrium solution but modifies the trajectory to reach it.

The acceleration of convergence option is used when *nn_acc*=1. In that case, $\Delta t = rn\_rdt$ is the time step of dynamics while $\widetilde{\Delta t} = rdttra$ is the tracer time-step. the former is set from the *rn_rdt* namelist parameter while the latter is computed using a hyperbolic tangent profile and the following namelist parameters : *rn_rdtmin*, *rn_rdtmax* and *rn_rdth*. Those three parameters correspond to the surface value the deep ocean value and the depth at which the transition occurs, respectively. The set of prognostic equations to solve

becomes :

$$\begin{aligned}\frac{\partial \mathbf{U}_h}{\partial t} &\equiv \frac{\mathbf{U}_h^{t+1} - \mathbf{U}_h^{t-1}}{2\Delta t} = \dots \\ \frac{\partial T}{\partial t} &\equiv \frac{T^{t+1} - T^{t-1}}{2\widetilde{\Delta t}} = \dots \\ \frac{\partial S}{\partial t} &\equiv \frac{S^{t+1} - S^{t-1}}{2\widetilde{\Delta t}} = \dots\end{aligned} \tag{14.1}$$

? has examined the consequences of this distorted physics. Free waves have a slower phase speed, their meridional structure is slightly modified, and the growth rate of baroclinically unstable waves is reduced but with a wider range of instability. This technique is efficient for searching for an equilibrium state in coarse resolution models. However its application is not suitable for many oceanic problems : it cannot be used for transient or time evolving problems (in particular, it is very questionable to use this technique when there is a seasonal cycle in the forcing fields), and it cannot be used in high-resolution models where baroclinically unstable processes are important. Moreover, the vertical variation of $\widetilde{\Delta t}$ implies that the heat and salt contents are no longer conserved due to the vertical coupling of the ocean level through both advection and diffusion. Therefore *rn_rdtmin = rn_rdtmax* should be a more clever choice.

14.5 Accuracy and Reproducibility (*lib_fortran.F90*)

14.5.1 Issues with intrinsinc SIGN function (key_nosignedzero)

The SIGN(A, B) is the FORTRAN intrinsic function delivers the magnitude of A with the sign of B. For example, SIGN(-3.0,2.0) has the value 3.0. The problematic case is when the second argument is zero, because, on platforms that support IEEE arithmetic, zero is actually a signed number. There is a positive zero and a negative zero.

In FORTRAN 90, the processor was required always to deliver a positive result for SIGN(A, B) if B was zero. Nevertheless, in FORTRAN 95, the processor is allowed to do the correct thing and deliver ABS(A) when B is a positive zero and -ABS(A) when B is a negative zero. This change in the specification becomes apparent only when B is of type real, and is zero, and the processor is capable of distinguishing between positive and negative zero, and B is negative real zero. Then SIGN delivers a negative result where, under FORTRAN 90 rules, it used to return a positive result. This change may be especially sensitive for the ice model, so we overwrite the intrinsinc function with our own function simply performing :

```
IF( B >= 0.e0 ) THEN   ;   SIGN(A,B) = ABS(A)
ELSE                   ;   SIGN(A,B) =-ABS(A)
ENDIF
```

This feature can be found in *lib_fortran.F90* module and is effective when **key_nosignedzero** is defined. We use a CPP key as the overwritting of a intrinsic function can present performance issues with some computers/compilers.

14.5.2 MPP reproducibility

The numerical reproducibility of simulations on distributed memory parallel computers is a critical issue. In particular, within NEMO global summation of distributed arrays is most susceptible to rounding errors, and their propagation and accumulation cause uncertainty in final simulation reproducibility on different numbers of processors. To avoid so, based on **?** review of different technics, we use a so called self-compensated summation method. The idea is to estimate the roundoff error, store it in a buffer, and then add it back in the next addition.

Suppose we need to calculate $b = a_1 + a_2 + a_3$. The following algorithm will allow to split the sum in two ($sum_1 = a_1 + a_2$ and $b = sum_2 = sum_1 + a_3$) with exactly the same rounding errors as the sum performed all at once.

$$
\begin{aligned}
sum_1 &= a_1 + a_2 \\
error_1 &= a_2 + (a_1 - sum_1) \\
sum_2 &= sum_1 + a_3 + error_1 \\
error_2 &= a_3 + error_1 + (sum_1 - sum_2) \\
b &= sum_2
\end{aligned}
$$

This feature can be found in *lib_fortran.F90* module and is effective when **key_mpp_rep**. In that case, all calls to glob_sum function (summation over the entire basin excluding duplicated rows and columns due to cyclic or north fold boundary condition as well as overlap MPP areas). Note this implementation may be sensitive to the optimization level.

14.6 Model Optimisation, Control Print and Benchmark

```
!-----------------------------------------------------------------------
&namctl        !   Control prints & Benchmark
!-----------------------------------------------------------------------
   ln_ctl      = .false.   !  trends control print (expensive!)
   nn_print    =    0      !  level of print (0 no extra print)
   nn_ictls    =    0      !  start i indice of control sum (use to compare mono versus
   nn_ictle    =    0      !  end   i indice of control sum        multi processor runs
   nn_jctls    =    0      !  start j indice of control               over a subdomain)
   nn_jctle    =    0      !  end   j indice of control
   nn_isplt    =    1      !  number of processors in i-direction
   nn_jsplt    =    1      !  number of processors in j-direction
   nn_bench    =    0      !  Bench mode (1/0): CAUTION use zero except for bench
                           !     (no physical validity of the results)
/
```

• Vector optimisation :

key_vectopt_loop enables the internal loops to collapse. This is very a very efficient way to increase the length of vector calculations and thus to speed up the model on vector computers.

• Control print

1- *ln_ctl* : compute and print the trends averaged over the interior domain in all TRA, DYN, LDF and ZDF modules. This option is very helpful when diagnosing the origin of an undesired change in model results.

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

3- **key_esopa** (to be rename key_nemo) : which is another option for model management. When defined, this key forces the activation of all options and CPP keys. For example, all tracer and momentum advection schemes are called ! Therefore the model results have no physical meaning. However, this option forces both the compiler and the model to run through all the FORTRAN lines of the model. This allows the user to check for obvious compilation or execution errors with all CPP options, and errors in namelist options.

4- last digit comparison (*nn_bit_cmp*). In an MPP simulation, the computation of a sum over the whole domain is performed as the summation over all processors of each of their sums over their interior domains. This double sum never gives exactly the same result as a single sum over the whole domain, due to truncation differences. The "bit comparison" option has been introduced in order to be able to check that mono-processor and multi-processor runs give exactly the same results.

• Benchmark (*nn_bench*). This option defines a benchmark run based on a GYRE configuration (see §**??**) in which the resolution remains the same whatever the domain size. This allows a very large model domain to be used, just by changing the domain size (*jpiglo*, *jpjglo*) and without adjusting either the time-step or the physical parameterisations.

14.7 Elliptic solvers (SOL)

```
!-----------------------------------------------------------------------
&namsol        !   elliptic solver / island / free surface
!-----------------------------------------------------------------------
   nn_solv     =      1    !  elliptic solver: =1 preconditioned conjugate gradient (pcg)
                           !                   =2 successive-over-relaxation (sor)
   nn_sol_arp  =      0    !  absolute/relative (0/1) precision convergence test
   rn_eps      =  1.e-6    !  absolute precision of the solver
   nn_nmin     =    300    !  minimum of iterations for the SOR solver
   nn_nmax     =    800    !  maximum of iterations for the SOR solver
   nn_nmod     =     10    !  frequency of test for the SOR solver
   rn_resmax   =  1.e-10   !  absolute precision for the SOR solver
   rn_sor      =  1.92     !  optimal coefficient for SOR solver (to be adjusted with the domain)
/
```

When the filtered sea surface height option is used, the surface pressure gradient is computed in *dynspg_flt.F90*. The force added in the momentum equation is solved implicitely. It is thus solution of an elliptic equation (2.6) for which two solvers are available : a Successive-Over-Relaxation scheme (SOR) and a preconditioned conjugate gradient scheme(PCG) [**??**]. The solver is selected trough the the value of *nn_solv* (namelist parameter).

The PCG is a very efficient method for solving elliptic equations on vector computers. It is a fast and rather easy method to use ; which are attractive features for a large number of ocean situations (variable bottom topography, complex coastal geometry, variable grid spacing, open or cyclic boundaries, etc ...). It does not require a search for an optimal parameter as in the SOR method. However, the SOR has been retained because it is a linear solver, which is a very useful property when using the adjoint model of *NEMO* .

At each time step, the time derivative of the sea surface height at time step $t+1$ (or equivalently the divergence of the *after* barotropic transport) that appears in the filtering forced is the solution of the elliptic equation obtained from the horizontal divergence of the vertical summation of (2.6). Introducing the following coefficients :

$$\begin{aligned} c_{i,j}^{NS} &= 2\Delta t^2 \, \frac{H_v(i,j)\; e_{1v}(i,j)}{e_{2v}(i,j)} \\ c_{i,j}^{EW} &= 2\Delta t^2 \, \frac{H_u(i,j)\; e_{2u}(i,j)}{e_{1u}(i,j)} \\ b_{i,j} &= \delta_i \left[e_{2u} M_u \right] - \delta_j \left[e_{1v} M_v \right] \;, \end{aligned} \tag{14.2}$$

the resulting five-point finite difference equation is given by :

$$\begin{aligned} c_{i+1,j}^{NS} D_{i+1,j} + \; c_{i,j+1}^{EW} D_{i,j+1} + c_{i,j}^{NS} D_{i-1,j} + \; c_{i,j}^{EW} D_{i,j-1} \\ - \left(c_{i+1,j}^{NS} + c_{i,j+1}^{EW} + c_{i,j}^{NS} + c_{i,j}^{EW} \right) D_{i,j} = b_{i,j} \end{aligned} \tag{14.3}$$

(13.3) 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 (13.3), is a vector.

Note that in the linear free surface case, the depth that appears in (13.2) does not vary with time, and thus the matrix can be computed once for all. In non-linear free surface (**key_vvl** defined) the matrix have to be updated at each time step.

14.7.1 Successive Over Relaxation (*nn_solv*=2, *solsor.F90*)

Let us introduce the four cardinal coefficients :

$$\begin{aligned} a_{i,j}^{S} &= c_{i,j}^{NS} / d_{i,j} & \qquad a_{i,j}^{W} &= c_{i,j}^{EW} / d_{i,j} \\ a_{i,j}^{E} &= c_{i,j+1}^{EW} / d_{i,j} & \qquad a_{i,j}^{N} &= c_{i+1,j}^{NS} / d_{i,j} \end{aligned}$$

where $d_{i,j} = c_{i,j}^{NS} + c_{i+1,j}^{NS} + c_{i,j}^{EW} + c_{i,j+1}^{EW}$ (i.e. the diagonal of the matrix). (13.3) can be rewritten as :

$$a_{i,j}^{N} D_{i+1,j} + a_{i,j}^{E} D_{i,j+1} + a_{i,j}^{S} D_{i-1,j} + a_{i,j}^{W} D_{i,j-1} - D_{i,j} = \tilde{b}_{i,j} \tag{14.4}$$

with $\tilde{b}_{i,j} = b_{i,j} / d_{i,j}$. (13.4) is the equation actually solved with the SOR method. This method used is an iterative one. Its algorithm can be summarised as follows (see **?** for a further discussion) :

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

$$D_{i,j}^{0} = 2\, D_{i,j}^{t} - D_{i,j}^{t-1} \tag{14.5}$$

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

$$\begin{aligned} R_{i,j}^{n} &= a_{i,j}^{N} D_{i+1,j}^{n} + a_{i,j}^{E} D_{i,j+1}^{n} + a_{i,j}^{S} D_{i-1,j}^{n+1} + a_{i,j}^{W} D_{i,j-1}^{n+1} - D_{i,j}^{n} - \tilde{b}_{i,j} \\ D_{i,j}^{n+1} &= D_{i,j}^{n} + \omega \; R_{i,j}^{n} \end{aligned} \tag{14.6}$$

where ω satisfies $1 \leq \omega \leq 2$. An optimal value exists for ω which significantly accelerates the convergence, but it has to be adjusted empirically for each model domain (except for a uniform grid where an analytical expression for ω can be found [**?**]). The value of ω is set using *rn_sor*, a **namelist** parameter. The convergence test is of the form :

$$\delta = \frac{\sum_{i,j} R_{i,j}^{n} R_{i,j}^{n}}{\sum_{i,j} \tilde{b}_{i,j}^{n} \tilde{b}_{i,j}^{n}} \leq \epsilon \tag{14.7}$$

where ϵ is the absolute precision that is required. It is recommended that a value smaller or equal to 10^{-6} is used for ϵ since larger values may lead to numerically induced basin scale barotropic oscillations. The precision is specified by setting *rn_eps* (**namelist** parameter). In addition, two other tests are used to halt the iterative algorithm. They involve the number of iterations and the modulus of the right hand side. If the former exceeds a specified value, *nn_max* (**namelist** parameter), or the latter is greater than 10^{15}, the whole model computation is stopped and the last computed time step fields are saved in a abort.nc NetCDF file. In both cases, this usually indicates that there is something wrong in the model configuration (an error in the mesh, the initial state, the input forcing, or the magnitude of the time step or of the mixing coefficients). A typical value of nn_max is a few hundred when $\epsilon = 10^{-6}$, increasing to a few thousand when $\epsilon = 10^{-12}$. The vectorization of the SOR algorithm is not straightforward. The scheme contains two linear recurrences on i and j. This inhibits the vectorisation. (13.6) can be been rewritten as :

$$\begin{aligned} R_{i,j}^{n} &= a_{i,j}^{N} D_{i+1,j}^{n} + a_{i,j}^{E} D_{i,j+1}^{n} + a_{i,j}^{S} D_{i-1,j}^{n} + {}_{i,j}^{W} D_{i,j-1}^{n} - D_{i,j}^{n} - \tilde{b}_{i,j} \\ R_{i,j}^{n} &= R_{i,j}^{n} - \omega \; a_{i,j}^{S} \; R_{i,j-1}^{n} \\ R_{i,j}^{n} &= R_{i,j}^{n} - \omega \; a_{i,j}^{W} \; R_{i-1,j}^{n} \end{aligned} \tag{14.8}$$

This technique slightly increases the number of iteration required to reach the convergence, but this is largely compensated by the gain obtained by the suppression of the recurrences.

Another technique have been chosen, the so-called red-black SOR. It consist in solving successively (13.6) for odd and even grid points. It also slightly reduced the convergence rate but allows the vectorisation. In addition, and this is the reason why it has been chosen, it is able to handle the north fold boundary condition used in ORCA configuration ($i.e.$ tri-polar global ocean mesh).

The SOR method is very flexible and can be used under a wide range of conditions, including irregular boundaries, interior boundary points, etc. Proofs of convergence, etc. may be found in the standard numerical methods texts for partial differential equations.

14.7.2 Preconditioned Conjugate Gradient (*nn_solv*=1, *solpcg.F90*)

A is a definite positive symmetric matrix, thus solving the linear system (13.3) is equivalent to the minimisation of a quadratic functional :

$$\mathbf{Ax} = \mathbf{b} \leftrightarrow \mathbf{x} = \inf_y \phi(\mathbf{y}) \quad , \qquad \phi(\mathbf{y}) = 1/2\langle \mathbf{Ay}, \mathbf{y}\rangle - \langle \mathbf{b}, \mathbf{y}\rangle$$

where $\langle , \rangle$ is the canonical dot product. The idea of the conjugate gradient method is to search for the solution in the following iterative way : assuming that $\mathbf{x}^n$ has been obtained, $\mathbf{x}^{n+1}$ is found from $\mathbf{x}^{n+1} = \mathbf{x}^n + \alpha^n \mathbf{d}^n$ which satisfies :

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

and expressing $\phi(\mathbf{y})$ as a function of α, we obtain the value that minimises the functional :

$$\alpha^n = \langle \mathbf{r}^n, \mathbf{r}^n \rangle / \langle\ \mathbf{A}\ \mathbf{d}^n, \mathbf{d}^n \rangle$$

where $\mathbf{r}^n = \mathbf{b} - \mathbf{A}\ \mathbf{x}^n = \mathbf{A}(\mathbf{x} - \mathbf{x}^n)$ is the error at rank n. The descent vector $\mathbf{d}^n$ s chosen to be dependent on the error : $\mathbf{d}^n = \mathbf{r}^n + \beta^n\ \mathbf{d}^{n-1}$. β^n is searched such that the descent vectors form an orthogonal basis for the dot product linked to **A**. Expressing the condition $\langle \mathbf{A}\ \mathbf{d}^n, \mathbf{d}^{n-1} \rangle = 0$ the value of β^n is found : $\beta^n = \langle \mathbf{r}^n, \mathbf{r}^n \rangle / \langle \mathbf{r}^{n-1}, \mathbf{r}^{n-1} \rangle$. As a result, the errors $\mathbf{r}^n$ form an orthogonal base for the canonic dot product while the descent vectors $\mathbf{d}^n$ form an orthogonal base for the dot product linked to **A**. The resulting algorithm is thus the following one :

initialisation :

$$\begin{aligned}
\mathbf{x}^0 &= D^0_{i,j} = 2D^t_{i,j} - D^{t-1}_{i,j} \quad \text{, the initial guess} \\
\mathbf{r}^0 &= \mathbf{d}^0 = \mathbf{b} - \mathbf{A}\ \mathbf{x}^0 \\
\gamma_0 &= \langle \mathbf{r}^0, \mathbf{r}^0 \rangle
\end{aligned}$$

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

$$\begin{aligned}
\mathbf{z}^n &= \mathbf{A}\ \mathbf{d}^n \\
\alpha_n &= \gamma_n / \langle \mathbf{z}^n, \mathbf{d}^n \rangle \\
\mathbf{x}^{n+1} &= \mathbf{x}^n + \alpha_n\ \mathbf{d}^n \\
\mathbf{r}^{n+1} &= \mathbf{r}^n - \alpha_n\ \mathbf{z}^n \\
\gamma_{n+1} &= \langle \mathbf{r}^{n+1}, \mathbf{r}^{n+1} \rangle \\
\beta_{n+1} &= \gamma_{n+1} / \gamma_n \\
\mathbf{d}^{n+1} &= \mathbf{r}^{n+1} + \beta_{n+1}\ \mathbf{d}^n
\end{aligned} \tag{14.9}$$

The convergence test is :

$$\delta = \gamma_n\ / \langle \mathbf{b}, \mathbf{b} \rangle \leq \epsilon \tag{14.10}$$

where ϵ is the absolute precision that is required. As for the SOR algorithm, the whole model computation is stopped when the number of iterations, *nn_max*, or the modulus of

the right hand side of the convergence equation exceeds a specified value (see §13.7.1 for a further discussion). The required precision and the maximum number of iterations allowed are specified by setting *rn_eps* and *nn_max* (**namelist** parameters).

It can be demonstrated that the above algorithm is optimal, provides the exact solution in a number of iterations equal to the size of the matrix, and that the convergence rate is faster as the matrix is closer to the identity matrix, $i.e.$ its eigenvalues are closer to 1. Therefore, it is more efficient to solve a better conditioned system which has the same solution. For that purpose, we introduce a preconditioning matrix $\mathbf{Q}$ which is an approximation of $\mathbf{A}$ but much easier to invert than $\mathbf{A}$, and solve the system :

$$\mathbf{Q}^{-1}\mathbf{A}\,\mathbf{x} = \mathbf{Q}^{-1}\mathbf{b} \tag{14.11}$$

The same algorithm can be used to solve (13.11) 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 $\mathbf{b}$ and $\mathbf{A}$ [**?**]. In *NEMO* , $\mathbf{Q}$ is chosen as the diagonal of $\mathbf{A}$, i.e. the simplest form for $\mathbf{Q}$ so that it can be easily inverted. In this case, the discrete formulation of (13.11) is in fact given by (13.4) and thus the matrix and right hand side are computed independently from the solver used.

15 Configurations

Contents

15.1 Introduction

The purpose of this part of the manual is to introduce the *NEMO* predefined configuration. These configurations are offered as means to explore various numerical and physical options, thus allowing the user to verify that the code is performing in a manner consistent with that we are running. This form of verification is critical as one adopts the code for his or her particular research purposes. The test cases also provide a sense for some of the options available in the code, though by no means are all options exercised in the predefined configurations.

15.2 Water column model : 1D model (C1D) (key_c1d)

The 1D model option simulates a stand alone water column within the 3D *NEMO* system. It can be applied to the ocean alone or to the ocean-ice system and can include passive tracers or a biogeochemical model. It is set up by defining the **key_c1d** CPP key. The 1D model is a very useful tool *(a)* to learn about the physics and numerical treatment of vertical mixing processes ; *(b)* to investigate suitable parameterisations of unresolved turbulence (surface wave breaking, Langmuir circulation, ...) ; *(c)* to compare the behaviour of different vertical mixing schemes ; *(d)* to perform sensitivity studies on the vertical diffusion at a particular point of an ocean domain ; *(d)* to produce extra diagnostics, without the large memory requirement of the full 3D model.

The methodology is based on the use of the zoom functionality over the smallest possible domain : a 3 x 3 domain centred on the grid point of interest (see §13.3), with some extra routines. There is no need to define a new mesh, bathymetry, initial state or forcing, since the 1D model will use those of the configuration it is a zoom of. The chosen grid point is set in par_oce.F90 module by setting the *jpizoom* and *jpjzoom* parameters to the indices of the location of the chosen grid point.

The 1D model has some specifies. First, all the horizontal derivatives are assumed to be zero. Therefore a simplified *step* routine is used (*step_c1d*) in which both lateral tendancy terms and lateral physics are not called, and the vertical velocity is zero (so far, no attempt at introducing a Ekman pumping velocity has been made). Second, the two components of the velocity are moved on a T-point. This requires a specific treatment of the Coriolis term (see *dyncor_c1d*) and of the dynamic time stepping (*dynnxt_c1d*). All the relevant modules can be found in the NEMOGCM/NEMO/OPA_SRC/C1D directory of the *NEMO* distribution.

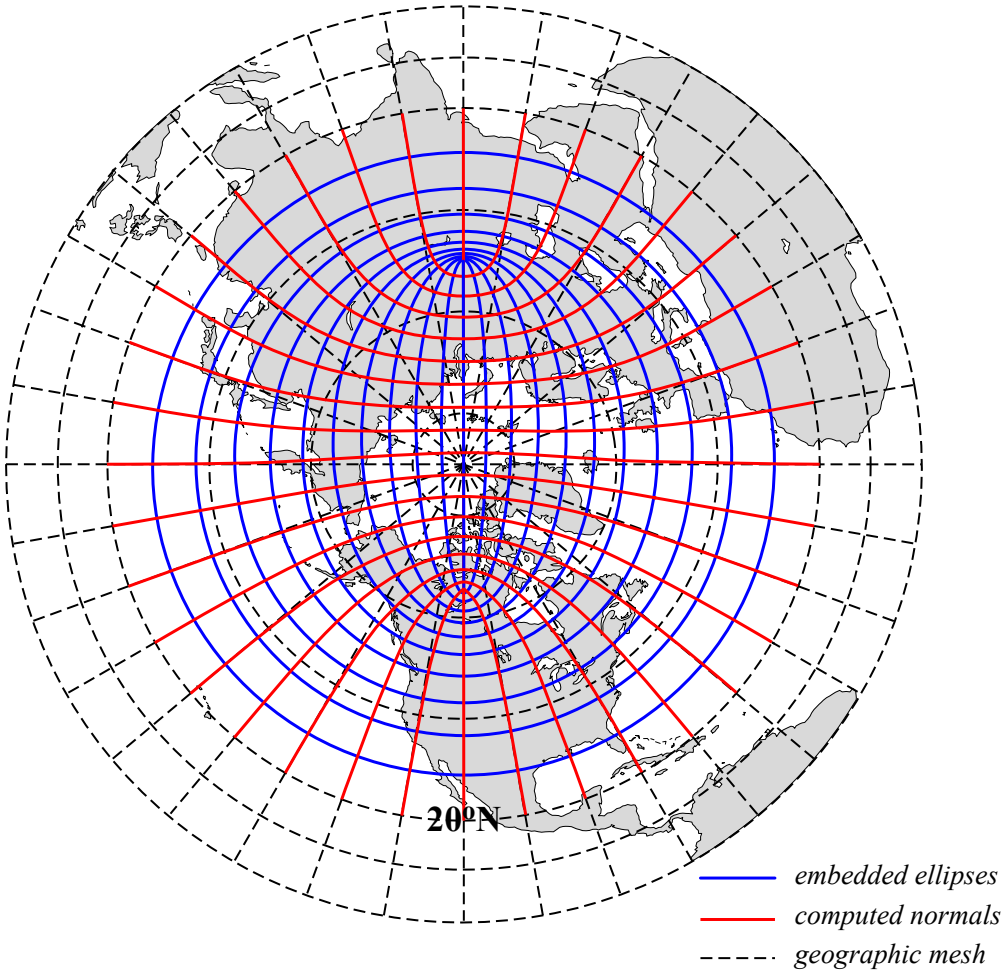

FIG. 15.1 – ORCA mesh conception. The departure from an isotropic Mercator grid start poleward of 20°N. The two "north pole" are the foci of a series of embedded ellipses (blue curves) which are determined analytically and form the i-lines of the ORCA mesh (pseudo latitudes). Then, following **?**, the normal to the series of ellipses (red curves) is computed which provide the j-lines of the mesh (pseudo longitudes).

15.3 ORCA family : global ocean with tripolar grid (key_orca_rX)

The ORCA family is a series of global ocean configurations that are run together with the LIM sea-ice model (ORCA-LIM) and possibly with PISCES biogeochemical model (ORCA-LIM-PISCES), using various resolutions.

CPP key	*jp_cfg*	*jpiglo*	*jpiglo*
key_orca_r4	4	92	76
key_orca_r2	2	182	149
key_orca_r1	1	362	292
key_orca_r05	05	722	511
key_orca_r025	025	1442	1021

TAB. 15.1 – Set of predefined parameters for ORCA family configurations. In all cases, the name of the configuration is set to "orca" ($i.e.$ *cp_cfg* = orca).

15.3.1 ORCA tripolar grid

The ORCA grid is a tripolar is based on the semi-analytical method of **?**. It allows to construct a global orthogonal curvilinear ocean mesh which has no singularity point inside the computational domain since two north mesh poles are introduced and placed on lands. The method involves defining an analytical set of mesh parallels in the stereographic polar plan, computing the associated set of mesh meridians, and projecting the resulting mesh onto the sphere. The set of mesh parallels used is a series of embedded ellipses which foci are the two mesh north poles (Fig. 14.1). The resulting mesh presents no loss of continuity in either the mesh lines or the scale factors, or even the scale factor derivatives over the whole ocean domain, as the mesh is not a composite mesh.

The method is applied to Mercator grid ($i.e.$ same zonal and meridional grid spacing) poleward of 20°N, so that the Equator is a mesh line, which provides a better numerical solution for equatorial dynamics. The choice of the series of embedded ellipses (position of the foci and variation of the ellipses) is a compromise between maintaining the ratio of mesh anisotropy (e_1/e_2) close to one in the ocean (especially in area of strong eddy activities such as the Gulf Stream) and keeping the smallest scale factor in the northern hemisphere larger than the smallest one in the southern hemisphere. The resulting mesh is shown in Fig. 14.1 and 14.2 for a half a degree grid (ORCA_R05). The smallest ocean scale factor is found in along Antarctica, while the ratio of anisotropy remains close to one except near the Victoria Island in the Canadian Archipelago.

15.3.2 ORCA pre-defined resolution

The NEMO system is provided with five built-in ORCA configurations which differ in the horizontal resolution. The value of the resolution is given by the resolution at the Equator expressed in degrees. Each of configuration is set through a CPP key, **key_orca_rX** (with X being an indicator of the resolution), which set the grid size and configuration name parameters (Tab. 14.1). .

The ORCA_R2 configuration has the following specificity : starting from a 2° ORCA mesh, local mesh refinements were applied to the Mediterranean, Red, Black and Caspian Seas, so that the resolution is 1°° there. A local transformation were also applied with in

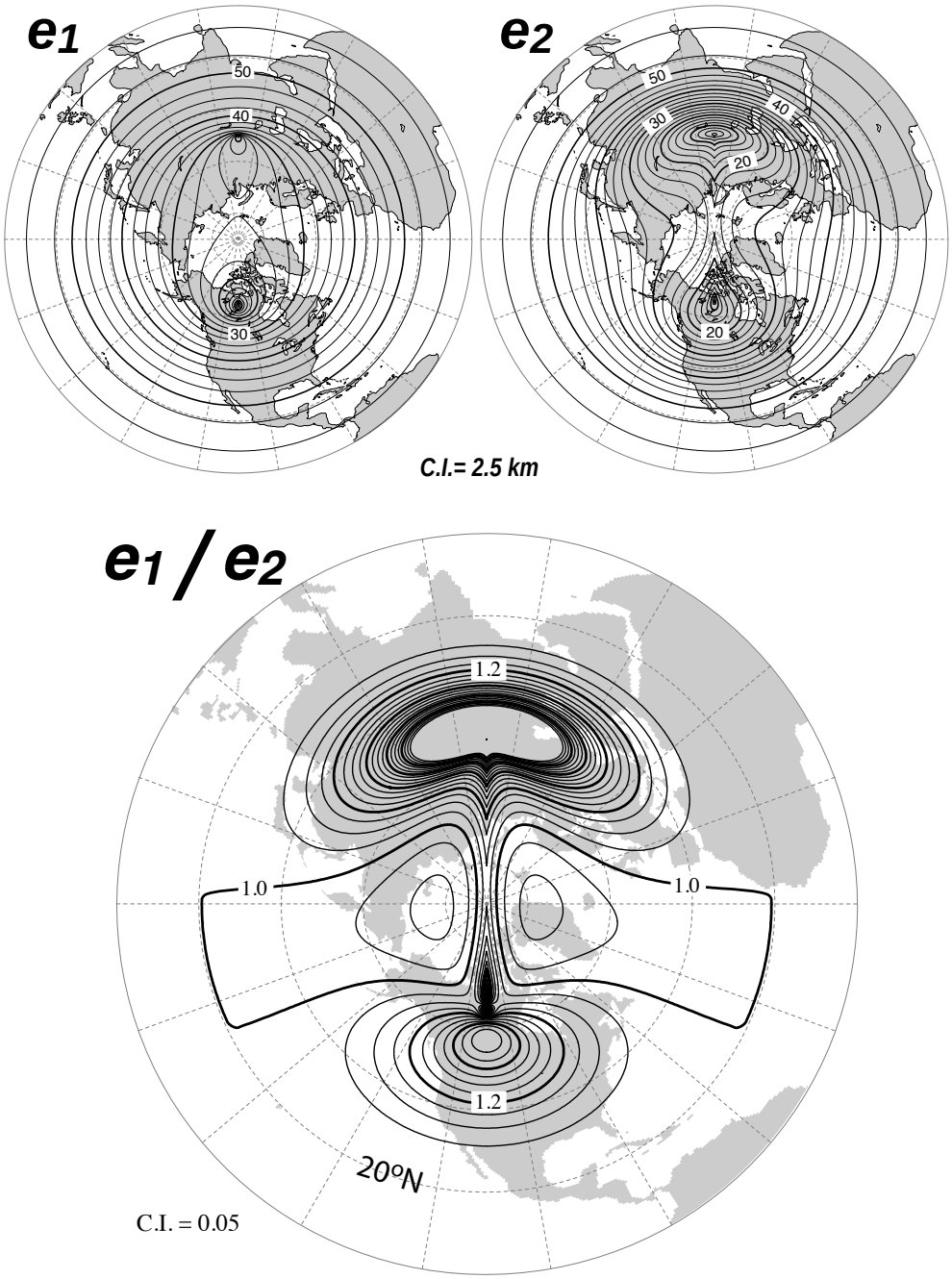

FIG. 15.2 – *Top* : Horizontal scale factors (e_1, e_2) and *Bottom* : ratio of anisotropy (e_1/e_2) for ORCA 0.5° mesh. South of 20°N a Mercator grid is used ($e_1 = e_2$) so that the anisotropy ratio is 1. Poleward of 20°N, the two "north pole" introduce a weak anisotropy over the ocean areas (< 1.2) except in vicinity of Victoria Island (Canadian Arctic Archipelago).

the Tropics in order to refine the meridional resolution up to 0.5˚at the Equator.

The ORCA_R1 configuration has only a local tropical transformation to refine the meridional resolution up to 1/3˚ at the Equator. Note that the tropical mesh refinements in ORCA_R2 and R1 strongly increases the mesh anisotropy there.

The ORCA_R05 and higher global configurations do not incorporate any regional refinements.

For ORCA_R1 and R025, setting the configuration key to 75 allows to use 75 vertical levels, otherwise 46 are used. In the other ORCA configurations, 31 levels are used (see Tab. 4.2 and Fig. 4.6).

Only the ORCA_R2 is provided with all its input files in the *NEMO* distribution. It is very similar to that used as part of the climate model developed at IPSL for the 4th IPCC assessment of climate change (Marti et al., 2009). It is also the basis for the *NEMO* contribution to the Coordinate Ocean-ice Reference Experiments (COREs) documented in **?**.

This version of ORCA_R2 has 31 levels in the vertical, with the highest resolution (10m) in the upper 150m (see Tab. 4.2 and Fig. 4.6). The bottom topography and the coastlines are derived from the global atlas of Smith and Sandwell (1997). The default forcing employ the boundary forcing from **?** (see §7.5.1), which was developed for the purpose of running global coupled ocean-ice simulations without an interactive atmosphere. This **?** dataset is available through the GFDL web site. The "normal year" of **?** has been chosen of the *NEMO* distribution since release v3.3.

ORCA_R2 pre-defined configuration can also be run with an AGRIF zoom over the Agulhas current area (**key_agrif** defined) and, by setting the key **key_arctic** or **key_antarctic**, a regional Arctic or peri-Antarctic configuration is extracted from an ORCA_R2 or R05 configurations using sponge layers at open boundaries.

15.4 GYRE family : double gyre basin (key_gyre)

The GYRE configuration [**?**] have been built to simulated the seasonal cycle of a double-gyre box model. It consist in an idealized domain similar to that used in the studies of **?** and **????**, over which an analytical seasonal forcing is applied. This allows to investigate the spontaneous generation of a large number of interacting, transient mesoscale eddies and their contribution to the large scale circulation.

The domain geometry is a closed rectangular basin on the β-plane centred at ~ 30˚N and rotated by 45˚, 3180 km long, 2120 km wide and 4 km deep (Fig. 13.1). The domain is bounded by vertical walls and by a ßat bottom. The configuration is meant to represent an idealized North Atlantic or North Pacific basin. The circulation is forced by analytical profiles of wind and buoyancy ßuxes. The applied forcings vary seasonally in a sinusoidal manner between winter and summer extrema [**?**]. The wind stress is zonal and its curl changes sign at 22˚N and 36˚N. It forces a subpolar gyre in the north, a subtropical gyre in the wider part of the domain and a small recirculation gyre in the southern corner. The net heat ßux takes the form of a restoring toward a zonal apparent air temperature profile.

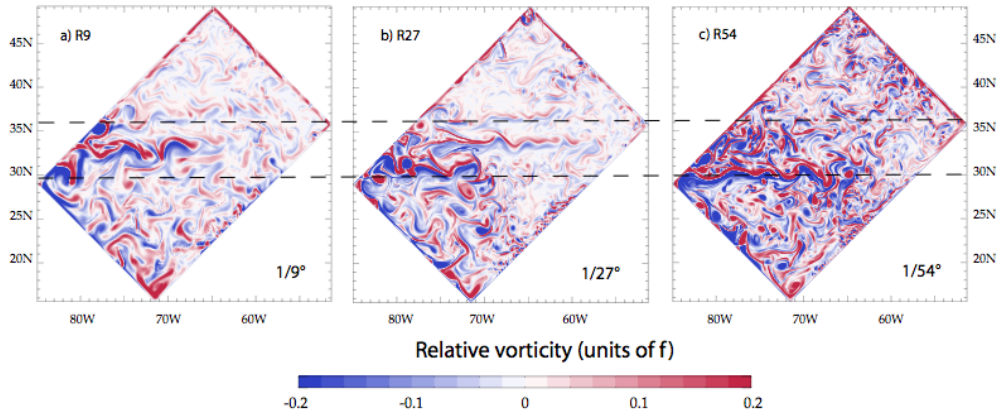

FIG. 15.3 – Snapshot of relative vorticity at the surface of the model domain in GYRE R9, R27 and R54. From **?**.

A portion of the net heat ßux which comes from the solar radiation is allowed to penetrate within the water column. The fresh water ßux is also prescribed and varies zonally. It is determined such as, at each time step, the basin-integrated ßux is zero. The basin is initialised at rest with vertical profiles of temperature and salinity uniformly applied to the whole domain.

The GYRE configuration is set through the **key_gyre** CPP key. Its horizontal resolution (and thus the size of the domain) is determined by setting *jp_cfg* in *par_GYRE.h90* file :
$jpiglo = 30 \times jp\_cfg + 2$
$jpjglo = 20 \times jp\_cfg + 2$
Obviously, the namelist parameters have to be adjusted to the chosen resolution. In the vertical, GYRE uses the default 30 ocean levels (*jpk*=31) (Fig. 4.6).

The GYRE configuration is also used in benchmark test as it is very simple to increase its resolution and as it does not requires any input file. For example, keeping a same model size on each processor while increasing the number of processor used is very easy, even though the physical integrity of the solution can be compromised.

15.5 EEL family : periodic channel

key_eel_r2 to be described....

key_eel_r5

key_eel_r6

15.6 POMME : mid-latitude sub-domain

key_pomme_r025 : to be described....

A Curvilinear $s-$Coordinate Equations

Contents

A.1 Chain rule of $s-$coordinate

In order to establish the set of Primitive Equation in curvilinear $s-$coordinates ($i.e.$ an orthogonal curvilinear coordinate in the horizontal and an Arbitrary Lagrangian Eulerian (ALE) coordinate in the vertical), we start from the set of equations established in §2.3.2 for the special case $k = z$ and thus $e_3 = 1$, and we introduce an arbitrary vertical coordinate $a = a(i, j, z, t)$. Let us define a new vertical scale factor by $e_3 = \partial z/\partial s$ (which now depends on (i, j, z, t)) and the horizontal slope of $s-$surfaces by :

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

The chain rule to establish the model equations in the curvilinear $s-$coordinate system is :

$$\begin{aligned}
\left.\frac{\partial \bullet}{\partial t}\right|_z &= \left.\frac{\partial \bullet}{\partial t}\right|_s - \frac{\partial \bullet}{\partial s}\frac{\partial s}{\partial t} \\
\left.\frac{\partial \bullet}{\partial i}\right|_z &= \left.\frac{\partial \bullet}{\partial i}\right|_s - \frac{\partial \bullet}{\partial s}\frac{\partial s}{\partial i} = \left.\frac{\partial \bullet}{\partial i}\right|_s - \frac{e_1}{e_3}\sigma_1\frac{\partial \bullet}{\partial s} \\
\left.\frac{\partial \bullet}{\partial j}\right|_z &= \left.\frac{\partial \bullet}{\partial j}\right|_s - \frac{\partial \bullet}{\partial s}\frac{\partial s}{\partial j} = \left.\frac{\partial \bullet}{\partial j}\right|_s - \frac{e_2}{e_3}\sigma_2\frac{\partial \bullet}{\partial s} \\
\frac{\partial \bullet}{\partial z} &= \frac{1}{e_3}\frac{\partial \bullet}{\partial s}
\end{aligned} \tag{A.2}$$

In particular applying the time derivative chain rule to z provides the expression for w_s, the vertical velocity of the $s-$surfaces referenced to a fix z-coordinate :

$$w_s = \left.\frac{\partial z}{\partial t}\right|_s = \frac{\partial z}{\partial s}\frac{\partial s}{\partial t} = e_3\frac{\partial s}{\partial t} \tag{A.3}$$

A.2 Continuity Equation in $s-$coordinate

Using (A.2) and the fact that the horizontal scale factors e_1 and e_2 do not depend on the vertical coordinate, the divergence of the velocity relative to the (i,j,z) coordinate system is transformed as follows in order to obtain its expression in the curvilinear

$s-$coordinate system :

$$\begin{aligned}
\nabla \cdot \mathbf{U} &= \frac{1}{e_1\, e_2}\left[\left.\frac{\partial(e_2\, u)}{\partial i}\right|_z + \left.\frac{\partial(e_1\, v)}{\partial j}\right|_z\right] + \frac{\partial w}{\partial z} \\
&= \frac{1}{e_1\, e_2}\left[\left.\frac{\partial(e_2\, u)}{\partial i}\right|_s - \frac{e_1}{e_3}\sigma_1\frac{\partial(e_2\, u)}{\partial s} + \left.\frac{\partial(e_1\, v)}{\partial j}\right|_s - \frac{e_2}{e_3}\sigma_2\frac{\partial(e_1\, v)}{\partial s}\right] + \frac{\partial w}{\partial s}\,\frac{\partial s}{\partial z} \\
&= \frac{1}{e_1\, e_2}\left[\left.\frac{\partial(e_2\, u)}{\partial i}\right|_s + \left.\frac{\partial(e_1\, v)}{\partial j}\right|_s\right] + \frac{1}{e_3}\left[\frac{\partial w}{\partial s} - \sigma_1\frac{\partial u}{\partial s} - \sigma_2\frac{\partial v}{\partial s}\right] \\
&= \frac{1}{e_1\, e_2\, e_3}\left[\left.\frac{\partial(e_2\, e_3\, u)}{\partial i}\right|_s - e_2\, u\left.\frac{\partial e_3}{\partial i}\right|_s + \left.\frac{\partial(e_1\, e_3\, v)}{\partial j}\right|_s - e_1 v\left.\frac{\partial e_3}{\partial j}\right|_s\right] \\
&\qquad + \frac{1}{e_3}\left[\frac{\partial w}{\partial s} - \sigma_1\frac{\partial u}{\partial s} - \sigma_2\frac{\partial v}{\partial s}\right]
\end{aligned}$$

Noting that $\frac{1}{e_1}\left.\frac{\partial e_3}{\partial i}\right|_s = \frac{1}{e_1}\left.\frac{\partial^2 z}{\partial i\,\partial s}\right|_s = \frac{\partial}{\partial s}\left(\frac{1}{e_1}\left.\frac{\partial z}{\partial i}\right|_s\right) = \frac{\partial \sigma_1}{\partial s}$ and $\frac{1}{e_2}\left.\frac{\partial e_3}{\partial j}\right|_s = \frac{\partial \sigma_2}{\partial s}$, it becomes :

$$\begin{aligned}
\nabla \cdot \mathbf{U} &= \frac{1}{e_1\, e_2\, e_3}\left[\left.\frac{\partial(e_2\, e_3\, u)}{\partial i}\right|_s + \left.\frac{\partial(e_1\, e_3\, v)}{\partial j}\right|_s\right] \\
&\qquad + \frac{1}{e_3}\left[\frac{\partial w}{\partial s} - u\frac{\partial \sigma_1}{\partial s} - v\frac{\partial \sigma_2}{\partial s} - \sigma_1\frac{\partial u}{\partial s} - \sigma_2\frac{\partial v}{\partial s}\right] \\
&= \frac{1}{e_1\, e_2\, e_3}\left[\left.\frac{\partial(e_2\, e_3\, u)}{\partial i}\right|_s + \left.\frac{\partial(e_1\, e_3\, v)}{\partial j}\right|_s\right] + \frac{1}{e_3}\,\frac{\partial}{\partial s}\left[w - u\,\sigma_1 - v\,\sigma_2\right]
\end{aligned}$$

Here, w is the vertical velocity relative to the $z-$coordinate system. Introducing the dia-surface velocity component, ω, defined as the velocity relative to the moving $s-$surfaces and normal to them :

$$\omega = w - w_s - \sigma_1\, u - \sigma_2\, v \tag{A.5}$$

with w_s given by (A.3), we obtain the expression for the divergence of the velocity in the curvilinear $s-$coordinate system :

$$\begin{aligned}
\nabla \cdot \mathbf{U} &= \frac{1}{e_1\, e_2\, e_3}\left[\left.\frac{\partial(e_2\, e_3\, u)}{\partial i}\right|_s + \left.\frac{\partial(e_1\, e_3\, v)}{\partial j}\right|_s\right] + \frac{1}{e_3}\frac{\partial \omega}{\partial s} + \frac{1}{e_3}\frac{\partial w_s}{\partial s} \\
&= \frac{1}{e_1\, e_2\, e_3}\left[\left.\frac{\partial(e_2\, e_3\, u)}{\partial i}\right|_s + \left.\frac{\partial(e_1\, e_3\, v)}{\partial j}\right|_s\right] + \frac{1}{e_3}\frac{\partial \omega}{\partial s} + \frac{1}{e_3}\frac{\partial}{\partial s}\left(e_3\,\frac{\partial s}{\partial t}\right) \\
&= \frac{1}{e_1\, e_2\, e_3}\left[\left.\frac{\partial(e_2\, e_3\, u)}{\partial i}\right|_s + \left.\frac{\partial(e_1\, e_3\, v)}{\partial j}\right|_s\right] + \frac{1}{e_3}\frac{\partial \omega}{\partial s} + \frac{\partial}{\partial s}\frac{\partial s}{\partial t} + \frac{1}{e_3}\frac{\partial s}{\partial t}\frac{\partial e_3}{\partial s} \\
&= \frac{1}{e_1\, e_2\, e_3}\left[\left.\frac{\partial(e_2\, e_3\, u)}{\partial i}\right|_s + \left.\frac{\partial(e_1\, e_3\, v)}{\partial j}\right|_s\right] + \frac{1}{e_3}\frac{\partial \omega}{\partial s} + \frac{1}{e_3}\frac{\partial e_3}{\partial t}
\end{aligned}$$

As a result, the continuity equation (2.1c) in the $s-$coordinates is :

$$\frac{1}{e_3}\frac{\partial e_3}{\partial t} + \frac{1}{e_1\, e_2\, e_3}\left[\left.\frac{\partial(e_2\, e_3\, u)}{\partial i}\right|_s + \left.\frac{\partial(e_1\, e_3\, v)}{\partial j}\right|_s\right] + \frac{1}{e_3}\frac{\partial \omega}{\partial s} = 0 \tag{A.7}$$

A additional term has appeared that take into account the contribution of the time variation of the vertical coordinate to the volume budget.

A.3 Momentum Equation in $s-$coordinate

Here we only consider the first component of the momentum equation, the generalization to the second one being straightforward.

• Total derivative in vector invariant form

Let us consider (2.17), the first component of the momentum equation in the vector invariant form. Its total $z-$coordinate time derivative, $\frac{Du}{Dt}\big|_z$ can be transformed as follows in order to obtain its expression in the curvilinear $s-$coordinate system :

$$\begin{aligned}
\frac{Du}{Dt}\Big|_z &= \frac{\partial u}{\partial t}\Big|_z - \zeta|_z\, v + \frac{1}{2e_1}\,\frac{\partial(u^2+v^2)}{\partial i}\Big|_z + w\,\frac{\partial u}{\partial z} \\
&= \frac{\partial u}{\partial t}\Big|_z - \zeta|_z\, v + \frac{1}{e_1\,e_2}\left[\frac{\partial(e_2\,v)}{\partial i}\Big|_z - \frac{\partial(e_1\,u)}{\partial j}\Big|_z\right] v + \frac{1}{2e_1}\,\frac{\partial(u^2+v^2)}{\partial i}\Big|_z + w\,\frac{\partial u}{\partial z}
\end{aligned}$$

introducing the chain rule (A.2)

$$\begin{aligned}
= \frac{\partial u}{\partial t}\Big|_z &- \frac{1}{e_1\,e_2}\left[\frac{\partial(e_2\,v)}{\partial i}\Big|_s - \frac{\partial(e_1\,u)}{\partial j}\Big|_s - \frac{e_1}{e_3}\sigma_1\frac{\partial(e_2\,v)}{\partial s} + \frac{e_2}{e_3}\sigma_2\frac{\partial(e_1\,u)}{\partial s}\right] v \\
&+ \frac{1}{2e_1}\left(\frac{\partial(u^2+v^2)}{\partial i}\Big|_s - \frac{e_1}{e_3}\sigma_1\frac{\partial(u^2+v^2)}{\partial s}\right) + \frac{w}{e_3}\,\frac{\partial u}{\partial s}
\end{aligned}$$

$$\begin{aligned}
= \frac{\partial u}{\partial t}\Big|_z &+ \zeta|_s\; v + \frac{1}{2\,e_1}\,\frac{\partial(u^2+v^2)}{\partial i}\Big|_s \\
&+ \frac{w}{e_3}\,\frac{\partial u}{\partial s} - \left[\frac{\sigma_1}{e_3}\frac{\partial v}{\partial s} - \frac{\sigma_2}{e_3}\frac{\partial u}{\partial s}\right] v - \frac{\sigma_1}{2e_3}\frac{\partial(u^2+v^2)}{\partial s}
\end{aligned}$$

$$\begin{aligned}
= \frac{\partial u}{\partial t}\Big|_z &+ \zeta|_s\; v + \frac{1}{2\,e_1}\,\frac{\partial(u^2+v^2)}{\partial i}\Big|_s \\
&+ \frac{1}{e_3}\left[w\frac{\partial u}{\partial s} + \sigma_1 v\frac{\partial v}{\partial s} - \sigma_2 v\frac{\partial u}{\partial s} - \sigma_1 u\frac{\partial u}{\partial s} - \sigma_1 v\frac{\partial v}{\partial s}\right]
\end{aligned}$$

$$= \frac{\partial u}{\partial t}\Big|_z + \zeta|_s\; v + \frac{1}{2\,e_1}\,\frac{\partial(u^2+v^2)}{\partial i}\Big|_s + \frac{1}{e_3}\left[w - \sigma_2 v - \sigma_1 u\right]\,\frac{\partial u}{\partial s}$$

Introducing ω, the dia-a-surface velocity given by (A.5)

$$= \frac{\partial u}{\partial t}\Big|_z + \zeta|_s\; v + \frac{1}{2\,e_1}\,\frac{\partial(u^2+v^2)}{\partial i}\Big|_s + \frac{1}{e_3}\left(\omega - w_s\right)\frac{\partial u}{\partial s}$$

Applying the time derivative chain rule (first equation of (A.2)) to u and using (A.3) provides the expression of the last term of the right hand side,

$$w_s\,\frac{\partial u}{\partial s} = \frac{\partial s}{\partial t}\,\frac{\partial u}{\partial s} = \frac{\partial u}{\partial t}\Big|_s - \frac{\partial u}{\partial t}\Big|_z \quad ,$$

leads to the $s-$coordinate formulation of the total $z-$coordinate time derivative, $i.e.$ the total $s-$coordinate time derivative :

$$\frac{Du}{Dt}\Big|_s = \frac{\partial u}{\partial t}\Big|_s + \zeta|_s\; v + \frac{1}{2\,e_1}\,\frac{\partial(u^2+v^2)}{\partial i}\Big|_s + \frac{1}{e_3}\omega\,\frac{\partial u}{\partial s} \tag{A.9}$$

Therefore, the vector invariant form of the total time derivative has exactly the same mathematical form in $z-$ and $s-$coordinates. This is not the case for the flux form as shown in next paragraph.

• **Total derivative in flux form**

Let us start from the total time derivative in the curvilinear $s-$coordinate system we have just establish. Following the procedure used to establish (2.15), it can be transformed into :

$$\begin{aligned}
\frac{Du}{Dt}\Big|_s &= \frac{\partial u}{\partial t}\Big|_s \quad -\zeta\, v + \frac{1}{2\,e_1}\frac{\partial\left(u^2+v^2\right)}{\partial i} + \frac{1}{e_3}\omega\,\frac{\partial u}{\partial s} \\
&= \frac{\partial u}{\partial t}\Big|_s \quad +\frac{1}{e_1\,e_2}\left(\frac{\partial(e_2\,u\,u)}{\partial i} + \frac{\partial(e_1\,u\,v)}{\partial j}\right) + \frac{1}{e_3}\frac{\partial(\omega\,u)}{\partial s} \\
&\qquad\qquad - u\left[\frac{1}{e_1 e_2}\left(\frac{\partial(e_2 u)}{\partial i} + \frac{\partial(e_1 v)}{\partial j}\right) + \frac{1}{e_3}\frac{\partial \omega}{\partial s}\right] \\
&\qquad\qquad -\frac{v}{e_1 e_2}\left(v\,\frac{\partial e_2}{\partial i} - u\,\frac{\partial e_1}{\partial j}\right)
\end{aligned}$$

Introducing the vertical scale factor inside the horizontal derivative of the first two terms ($i.e.$ the horizontal divergence), it becomes :

$$\begin{aligned}
\frac{Du}{Dt}\Big|_s &= \frac{\partial u}{\partial t}\Big|_s \quad +\frac{1}{e_1\,e_2\,e_3}\left(\frac{\partial(e_2 e_3\,u^2)}{\partial i} + \frac{\partial(e_1 e_3\,uv)}{\partial j} - e_2 uu\frac{\partial e_3}{\partial i} - e_1 uv\frac{\partial e_3}{\partial j}\right) + \frac{1}{e_3}\frac{\partial(\omega\,u)}{\partial s} \\
&\qquad\qquad - u\left[\frac{1}{e_1 e_2 e_3}\left(\frac{\partial(e_2 e_3\,u)}{\partial i} + \frac{\partial(e_1 e_3\,v)}{\partial j} - e_2 u\,\frac{\partial e_3}{\partial i} - e_1 v\,\frac{\partial e_3}{\partial j}\right) - \frac{1}{e_3}\frac{\partial \omega}{\partial s}\right] \\
&\qquad\qquad -\frac{v}{e_1 e_2}\left(v\,\frac{\partial e_2}{\partial i} - u\,\frac{\partial e_1}{\partial j}\right) \\
&= \frac{\partial u}{\partial t}\Big|_s \quad +\frac{1}{e_1\,e_2\,e_3}\left(\frac{\partial(e_2 e_3\,u\,u)}{\partial i} + \frac{\partial(e_1 e_3\,u\,v)}{\partial j}\right) + \frac{1}{e_3}\frac{\partial(\omega\,u)}{\partial s} \\
&\qquad\qquad - u\left[\frac{1}{e_1 e_2 e_3}\left(\frac{\partial(e_2 e_3\,u)}{\partial i} + \frac{\partial(e_1 e_3\,v)}{\partial j}\right) - \frac{1}{e_3}\frac{\partial \omega}{\partial s}\right] - \frac{v}{e_1 e_2}\left(v\,\frac{\partial e_2}{\partial i} - u\,\frac{\partial e_1}{\partial j}\right)
\end{aligned}$$

Introducing a more compact form for the divergence of the momentum fluxes, and using (A.7), the $s-$coordinate continuity equation, it becomes :

$$= \frac{\partial u}{\partial t}\Big|_s \quad + \nabla\cdot(\mathbf{U}\,u)\big|_s + u\frac{1}{e_3}\frac{\partial e_3}{\partial t} - \frac{v}{e_1 e_2}\left(v\,\frac{\partial e_2}{\partial i} - u\,\frac{\partial e_1}{\partial j}\right)$$

which leads to the $s-$coordinate flux formulation of the total $s-$coordinate time derivative, $i.e.$ the total $s-$coordinate time derivative in flux form :

$$\frac{Du}{Dt}\bigg|_s = \frac{1}{e_3}\,\frac{\partial(e_3\,u)}{\partial t}\bigg|_s + \nabla\cdot(\mathbf{U}\,u)\big|_s - \frac{v}{e_1 e_2}\left(v\,\frac{\partial e_2}{\partial i} - u\,\frac{\partial e_1}{\partial j}\right) \tag{A.11}$$

which is the total time derivative expressed in the curvilinear $s-$coordinate system. It has the same form as in the $z-$coordinate but for the vertical scale factor that has appeared inside the time derivative which comes from the modification of (A.7), the continuity equation.

• horizontal pressure gradient

The horizontal pressure gradient term can be transformed as follows :

$$\begin{aligned} -\frac{1}{\rho_o\, e_1}\left.\frac{\partial p}{\partial i}\right|_z &= -\frac{1}{\rho_o e_1}\left[\left.\frac{\partial p}{\partial i}\right|_s - \frac{e_1}{e_3}\sigma_1\frac{\partial p}{\partial s}\right] \\ &= -\frac{1}{\rho_o\, e_1}\left.\frac{\partial p}{\partial i}\right|_s + \frac{\sigma_1}{\rho_o\, e_3}\,(-g\;\rho\;e_3) \\ &= -\frac{1}{\rho_o\, e_1}\left.\frac{\partial p}{\partial i}\right|_s - \frac{g\;\rho}{\rho_o}\sigma_1 \end{aligned}$$

Applying similar manipulation to the second component and replacing σ_1 and σ_2 by their expression (A.1), it comes :

$$\begin{aligned} -\frac{1}{\rho_o\, e_1}\left.\frac{\partial p}{\partial i}\right|_z &= -\frac{1}{\rho_o\, e_1}\left(\left.\frac{\partial p}{\partial i}\right|_s + g\;\rho\;\left.\frac{\partial z}{\partial i}\right|_s\right) \\ -\frac{1}{\rho_o\, e_2}\left.\frac{\partial p}{\partial j}\right|_z &= -\frac{1}{\rho_o\, e_2}\left(\left.\frac{\partial p}{\partial j}\right|_s + g\;\rho\;\left.\frac{\partial z}{\partial j}\right|_s\right) \end{aligned} \tag{A.12}$$

An additional term appears in (A.14) which accounts for the tilt of $s-$surfaces with respect to geopotential $z-$surfaces.

As in z-coordinate, the horizontal pressure gradient can be split in two parts following **?**. Let defined a density anomaly, d, by $d = (\rho - \rho_o)/\rho_o$, and a hydrostatic pressure anomaly, p'_h, by $p'_h = g\;\int_z^\eta d\;e_3\;dk$. The pressure is then given by :

$$\begin{aligned} p &= g\;\int_z^\eta \rho\;e_3\;dk = g\;\int_z^\eta (\rho_o\,d + 1)\;\;e_3\;dk \\ &= g\,\rho_o\;\int_z^\eta d\;e_3\;dk + g\;\int_z^\eta e_3\;dk \end{aligned}$$

Therefore, p and p'_h are linked through :

$$p = \rho_o\;p'_h + g\,(z+\eta) \tag{A.13}$$

and the hydrostatic pressure balance expressed in terms of p'_h and d is :

$$\frac{\partial p'_h}{\partial k} = -d\,g\,e_3$$

Substituing (A.13) in (A.14) and using the definition of the density anomaly it comes the expression in two parts :

$$\begin{aligned} -\frac{1}{\rho_o\, e_1}\left.\frac{\partial p}{\partial i}\right|_z &= -\frac{1}{e_1}\left(\left.\frac{\partial p'_h}{\partial i}\right|_s + g\, d\, \left.\frac{\partial z}{\partial i}\right|_s\right) - \frac{g}{e_1}\frac{\partial \eta}{\partial i} \\ -\frac{1}{\rho_o\, e_2}\left.\frac{\partial p}{\partial j}\right|_z &= -\frac{1}{e_2}\left(\left.\frac{\partial p'_h}{\partial j}\right|_s + g\, d\, \left.\frac{\partial z}{\partial j}\right|_s\right) - \frac{g}{e_2}\frac{\partial \eta}{\partial j} \end{aligned} \tag{A.14}$$

This formulation of the pressure gradient is characterised by the appearance of a term depending on the the sea surface height only (last term on the right hand side of expression (A.14)). This term will be abusively named *surface pressure gradient* whereas the first term will be named *hydrostatic pressure gradient* by analogy to the z-coordinate formulation. In fact, the the true surface pressure gradient is $1/\rho_o \nabla(\rho\eta)$, and η is implicitly included in the computation of p'_h through the upper bound of the vertical integration.

• The other terms of the momentum equation

The coriolis and forcing terms as well as the the vertical physics remain unchanged as they involve neither time nor space derivatives. The form of the lateral physics is discussed in appendix B.

• Full momentum equation

To sum up, in a curvilinear s-coordinate system, the vector invariant momentum equation solved by the model has the same mathematical expression as the one in a curvilinear $z-$coordinate, but the pressure gradient term :

$$\begin{aligned} \frac{\partial u}{\partial t} = &+ (\zeta + f)\, v - \frac{1}{2\, e_1}\frac{\partial}{\partial i}\left(u^2 + v^2\right) - \frac{1}{e_3}\omega\frac{\partial u}{\partial k} \\ &- \frac{1}{e_1}\left(\frac{\partial p'_h}{\partial i} + g\, d\, \frac{\partial z}{\partial i}\right) - \frac{g}{e_1}\frac{\partial \eta}{\partial i} + D_u^{\mathbf{U}} + F_u^{\mathbf{U}} \end{aligned} \tag{A.15a}$$

$$\begin{aligned} \frac{\partial v}{\partial t} = &- (\zeta + f)\, u - \frac{1}{2\, e_2}\frac{\partial}{\partial j}\left(u^2 + v^2\right) - \frac{1}{e_3}\omega\frac{\partial v}{\partial k} \\ &- \frac{1}{e_2}\left(\frac{\partial p'_h}{\partial j} + g\, d\, \frac{\partial z}{\partial j}\right) - \frac{g}{e_2}\frac{\partial \eta}{\partial j} + D_v^{\mathbf{U}} + F_v^{\mathbf{U}} \end{aligned} \tag{A.15b}$$

whereas the flux form momentum equation differ from it by the formulation of both the time derivative and the pressure gradient term :

$$\begin{aligned} \frac{1}{e_3}\frac{\partial\,(e_3\, u)}{\partial t} = &\nabla\cdot(\mathbf{U}\, u) + \left\{ f + \frac{1}{e_1 e_2}\left(v\,\frac{\partial e_2}{\partial i} - u\,\frac{\partial e_1}{\partial j}\right)\right\} v \\ &- \frac{1}{e_1}\left(\frac{\partial p'_h}{\partial i} + g\, d\, \frac{\partial z}{\partial i}\right) - \frac{g}{e_1}\frac{\partial \eta}{\partial i} + D_u^{\mathbf{U}} + F_u^{\mathbf{U}} \end{aligned} \tag{A.16a}$$

$$\frac{1}{e_3}\frac{\partial\left(e_3\,v\right)}{\partial t} = -\nabla\cdot\left(\mathbf{U}\,v\right) + \left\{ f + \frac{1}{e_1 e_2}\left(v\,\frac{\partial e_2}{\partial i} - u\,\frac{\partial e_1}{\partial j}\right)\right\} u \\ - \frac{1}{e_2}\left(\frac{\partial p'_h}{\partial j} + g\,d\,\frac{\partial z}{\partial j}\right) - \frac{g}{e_2}\frac{\partial \eta}{\partial j} + D_v^{\mathbf{U}} + F_v^{\mathbf{U}} \quad \text{(A.16b)}$$

Both formulation share the same hydrostatic pressure balance expressed in terms of hydrostatic pressure and density anmalies, p'_h and $d = (\frac{\rho}{\rho_o} - 1)$:

$$\frac{\partial p'_h}{\partial k} = -d\,g\,e_3 \quad \text{(A.17)}$$

It is important to realize that the change in coordinate system has only concerned the position on the vertical. It has not affected (**i**,**j**,**k**), the orthogonal curvilinear set of unit vector. (u,v) are always horizontal velocities so that their evolution is driven by *horizontal* forces, in particular the pressure gradient. By contrast, ω is not w, the third component of the velocity, but the dia-surface velocity component, $i.e.$ the velocity relative to the moving $s-$surfaces and normal to them.

A.4 Tracer Equation

The tracer equation is obtained using the same calculation as for the continuity equation and then regrouping the time derivative terms in the left hand side :

$$\frac{1}{e_3}\frac{\partial\left(e_3 T\right)}{\partial t} = -\frac{1}{e_1\,e_2\,e_3}\left[\frac{\partial}{\partial i}\left(e_2\,e_3\,Tu\right) + \frac{\partial}{\partial j}\left(e_1\,e_3\,Tv\right)\right] \\ + \frac{1}{e_3}\frac{\partial}{\partial k}\left(Tw\right) + D^T + F^T \quad \text{(A.18)}$$

The expression for the advection term is a straight consequence of (A.4), the expression of the 3D divergence in the $s-$coordinates established above.

B Appendix B : Diffusive Operators

Contents

B.1 Horizontal/Vertical 2nd Order Tracer Diffusive Operators

In the z-coordinate, the horizontal/vertical second order tracer diffusion operator is given by :

$$D^T = \frac{1}{e_1\,e_2}\left[\frac{\partial}{\partial i}\left(\frac{e_2}{e_1}A^{lT}\;\frac{\partial T}{\partial i}\Big|_z\right)\Big|_z + \frac{\partial}{\partial j}\left(\frac{e_1}{e_2}A^{lT}\;\frac{\partial T}{\partial j}\Big|_z\right)\Big|_z\right] + \frac{\partial}{\partial z}\left(A^{vT}\;\frac{\partial T}{\partial z}\right) \quad \text{(B.1)}$$

In the s-coordinate, we defined the slopes of s-surfaces, σ_1 and σ_2 by (A.1) and the vertical/horizontal ratio of diffusion coefficient by $\epsilon = A^{vT}/A^{lT}$. The diffusion operator is given by :

$$D^T = \nabla|_s \cdot \left[A^{lT}\;\Re \cdot \nabla|_s\, T\right] \quad \text{where } \Re = \begin{pmatrix} 1 & 0 & -\sigma_1 \\ 0 & 1 & -\sigma_2 \\ -\sigma_1 & -\sigma_2 & \varepsilon + \sigma_1^2 + \sigma_2^2 \end{pmatrix} \quad \text{(B.2)}$$

or in expanded form :

$$\begin{aligned} D^T = \frac{1}{e_1\,e_2\,e_3}\Bigg[& e_2\,e_3\,A^{lT}\;\frac{\partial}{\partial i}\left(\frac{1}{e_1}\;\frac{\partial T}{\partial i}\Big|_s - \frac{\sigma_1}{e_3}\;\frac{\partial T}{\partial s}\right)\Big|_s \\ & + e_1\,e_3\,A^{lT}\;\frac{\partial}{\partial j}\left(\frac{1}{e_2}\;\frac{\partial T}{\partial j}\Big|_s - \frac{\sigma_2}{e_3}\;\frac{\partial T}{\partial s}\right)\Big|_s \\ & + e_1\,e_2\,A^{lT}\;\frac{\partial}{\partial s}\left(-\frac{\sigma_1}{e_1}\;\frac{\partial T}{\partial i}\Big|_s - \frac{\sigma_2}{e_2}\;\frac{\partial T}{\partial j}\Big|_s + \left(\varepsilon+\sigma_1^2+\sigma_2^2\right)\;\frac{1}{e_3}\;\frac{\partial T}{\partial s}\right)\Bigg] \end{aligned}$$

Equation (B.2) is obtained from (B.1) without any additional assumption. Indeed, for the special case $k = z$ and thus $e_3 = 1$, we introduce an arbitrary vertical coordinate $s = s(i, j, z)$ as in Appendix A and use (A.1) and (A.2). Since no cross horizontal derivative $\partial_i\partial_j$ appears in (B.1), the (i,z) and (j,z) planes are independent. The derivation can then be demonstrated for the $(i,z) \to (j,s)$ transformation without any loss of generality :

$$\begin{aligned} D^T &= \frac{1}{e_1\,e_2}\;\frac{\partial}{\partial i}\left(\frac{e_2}{e_1}A^{lT}\;\frac{\partial T}{\partial i}\Big|_z\right)\Big|_z + \frac{\partial}{\partial z}\left(A^{vT}\;\frac{\partial T}{\partial z}\right) \\ &= \frac{1}{e_1\,e_2}\Bigg[\frac{\partial}{\partial i}\left(\frac{e_2}{e_1}A^{lT}\;\left(\frac{\partial T}{\partial i}\Big|_s - \frac{e_1\,\sigma_1}{e_3}\frac{\partial T}{\partial s}\right)\right)\Big|_s \\ &\qquad - \frac{e_1\,\sigma_1}{e_3}\frac{\partial}{\partial s}\left(\frac{e_2}{e_1}A^{lT}\;\left(\frac{\partial T}{\partial i}\Big|_s - \frac{e_1\,\sigma_1}{e_3}\frac{\partial T}{\partial s}\right)\Big|_s\right)\Bigg] + \frac{1}{e_3}\frac{\partial}{\partial s}\left[\frac{A^{vT}}{e_3}\;\frac{\partial T}{\partial s}\right] \\ &= \frac{1}{e_1\,e_2\,e_3}\Bigg[\frac{\partial}{\partial i}\left(\frac{e_2\,e_3}{e_1}A^{lT}\;\frac{\partial T}{\partial i}\Big|_s\right)\Big|_s - \frac{e_2}{e_1}A^{lT}\;\frac{\partial e_3}{\partial i}\Big|_s\;\frac{\partial T}{\partial i}\Big|_s \\ &\qquad - e_3\frac{\partial}{\partial i}\left(\frac{e_2\,\sigma_1}{e_3}A^{lT}\;\frac{\partial T}{\partial s}\right)\Big|_s - e_1\,\sigma_1\frac{\partial}{\partial s}\left(\frac{e_2}{e_1}A^{lT}\;\frac{\partial T}{\partial i}\Big|_s\right) \\ &\qquad - e_1\,\sigma_1\frac{\partial}{\partial s}\left(-\frac{e_2\,\sigma_1}{e_3}A^{lT}\;\frac{\partial T}{\partial s}\right) + \frac{\partial}{\partial s}\left(\frac{e_1\,e_2}{e_3}A^{vT}\;\frac{\partial T}{\partial s}\right)\Bigg] \end{aligned}$$

Noting that $\frac{1}{e_1}\ \frac{\partial e_3}{\partial i}\Big|_s = \frac{\partial \sigma_1}{\partial s}$, it becomes :

$$
\begin{aligned}
= \frac{1}{e_1\,e_2\,e_3}\Bigg[\quad & \frac{\partial}{\partial i}\left(\frac{e_2\,e_3}{e_1}\,A^{lT}\ \frac{\partial T}{\partial i}\Big|_s\right)\Big|_s - e_3\frac{\partial}{\partial i}\left(\frac{e_2\,\sigma_1}{e_3}A^{lT}\ \frac{\partial T}{\partial s}\right)\Big|_s \\
& -e_2 A^{lT}\ \frac{\partial \sigma_1}{\partial s}\ \frac{\partial T}{\partial i}\Big|_s - e_1\,\sigma_1\frac{\partial}{\partial s}\left(\frac{e_2}{e_1}A^{lT}\ \frac{\partial T}{\partial i}\Big|_s\right) \\
& +e_1\,\sigma_1\frac{\partial}{\partial s}\left(\frac{e_2\,\sigma_1}{e_3}A^{lT}\ \frac{\partial T}{\partial s}\right) + \frac{\partial}{\partial s}\left(\frac{e_1\,e_2}{e_3}A^{vT}\ \frac{\partial T}{\partial z}\right) \quad \Bigg]
\end{aligned}
$$

$$
\begin{aligned}
= \frac{1}{e_1\,e_2\,e_3}\Bigg[\quad & \frac{\partial}{\partial i}\left(\frac{e_2\,e_3}{e_1}A^{lT}\ \frac{\partial T}{\partial i}\Big|_s\right)\Big|_s - \frac{\partial}{\partial i}\left(e_2\,\sigma_1 A^{lT}\ \frac{\partial T}{\partial s}\right)\Big|_s \\
& +\frac{e_2\,\sigma_1}{e_3}A^{lT}\ \frac{\partial T}{\partial s}\ \frac{\partial e_3}{\partial i}\Big|_s - e_2 A^{lT}\ \frac{\partial \sigma_1}{\partial s}\ \frac{\partial T}{\partial i}\Big|_s \\
& -e_2\,\sigma_1\frac{\partial}{\partial s}\left(A^{lT}\ \frac{\partial T}{\partial i}\Big|_s\right) + \frac{\partial}{\partial s}\left(\frac{e_1\,e_2\,\sigma_1^2}{e_3}A^{lT}\ \frac{\partial T}{\partial s}\right) \\
& -\frac{\partial(e_1\,e_2\,\sigma_1)}{\partial s}\left(\frac{\sigma_1}{e_3}A^{lT}\ \frac{\partial T}{\partial s}\right) + \frac{\partial}{\partial s}\left(\frac{e_1\,e_2}{e_3}A^{vT}\ \frac{\partial T}{\partial s}\right) \quad \Bigg]
\end{aligned}
$$

using the same remark as just above, it becomes :

$$
\begin{aligned}
= \frac{1}{e_1\,e_2\,e_3}\Bigg[\quad & \frac{\partial}{\partial i}\left(\frac{e_2\,e_3}{e_1}A^{lT}\ \frac{\partial T}{\partial i}\Big|_s - e_2\,\sigma_1 A^{lT}\ \frac{\partial T}{\partial s}\right)\Big|_s \\
& +\frac{e_1\,e_2\,\sigma_1}{e_3}A^{lT}\ \frac{\partial T}{\partial s}\ \frac{\partial \sigma_1}{\partial s} - \frac{\sigma_1}{e_3}A^{lT}\ \frac{\partial(e_1\,e_2\,\sigma_1)}{\partial s}\ \frac{\partial T}{\partial s} \\
& -e_2\left(A^{lT}\ \frac{\partial \sigma_1}{\partial s}\ \frac{\partial T}{\partial i}\Big|_s + \frac{\partial}{\partial s}\left(\sigma_1 A^{lT}\ \frac{\partial T}{\partial i}\Big|_s\right) - \frac{\partial \sigma_1}{\partial s}\ A^{lT}\ \frac{\partial T}{\partial i}\Big|_s\right) \\
& +\frac{\partial}{\partial s}\left(\frac{e_1\,e_2\,\sigma_1^2}{e_3}A^{lT}\ \frac{\partial T}{\partial s} + \frac{e_1\,e_2}{e_3}A^{vT}\ \frac{\partial T}{\partial s}\right) \quad \Bigg]
\end{aligned}
$$

Since the horizontal scale factors do not depend on the vertical coordinate, the last term of the first line and the first term of the last line cancel, while the second line reduces to a single vertical derivative, so it becomes :

$$
\begin{aligned}
= \frac{1}{e_1\,e_2\,e_3}\Bigg[\quad & \frac{\partial}{\partial i}\left(\frac{e_2\,e_3}{e_1}A^{lT}\ \frac{\partial T}{\partial i}\Big|_s - e_2\,\sigma_1\,A^{lT}\ \frac{\partial T}{\partial s}\right)\Big|_s \\
& +\frac{\partial}{\partial s}\left(-e_2\,\sigma_1\,A^{lT}\ \frac{\partial T}{\partial i}\Big|_s + A^{lT}\,\frac{e_1\,e_2}{e_3}\ \left(\varepsilon + \sigma_1^2\right)\frac{\partial T}{\partial s}\right) \quad \Bigg]
\end{aligned}
$$

in other words, the horizontal Laplacian operator in the (i,s) plane takes the following form :

$$
D^T = \frac{1}{e_1\,e_2\,e_3}\begin{pmatrix} \frac{\partial(e_2 e_3 \bullet)}{\partial i}\Big|_s \\ \frac{\partial(e_1 e_2 \bullet)}{\partial s}\end{pmatrix} \cdot \left[A^{lT}\begin{pmatrix} 1 & -\sigma_1 \\ -\sigma_1 & \varepsilon_1^2\end{pmatrix} \cdot \begin{pmatrix} \frac{1}{e_1}\ \frac{\partial \bullet}{\partial i}\Big|_s \\ \frac{1}{e_3}\ \frac{\partial \bullet}{\partial s}\end{pmatrix}(T)\right]
$$

B.2 Iso/diapycnal 2nd Order Tracer Diffusive Operators

The iso/diapycnal diffusive tensor $\mathbf{A_I}$ expressed in the (i,j,k) curvilinear coordinate system in which the equations of the ocean circulation model are formulated, takes the following form [**?**] :

$$\mathbf{A_I} = \frac{A^{lT}}{\left(1+a_1^2+a_2^2\right)} \begin{bmatrix} 1+a_1^2 & -a_1 a_2 & -a_1 \\ -a_1 a_2 & 1+a_2^2 & -a_2 \\ -a_1 & -a_2 & \varepsilon + a_1^2 + a_2^2 \end{bmatrix}$$

where (a_1, a_2) are the isopycnal slopes in (**i**, **j**) directions :

$$a_1 = \frac{e_3}{e_1}\left(\frac{\partial \rho}{\partial i}\right)\left(\frac{\partial \rho}{\partial k}\right)^{-1} \quad , \quad a_2 = \frac{e_3}{e_2}\left(\frac{\partial \rho}{\partial j}\right)\left(\frac{\partial \rho}{\partial k}\right)^{-1}$$

In practice, the isopycnal slopes are generally less than 10^{-2} in the ocean, so $\mathbf{A_I}$ can be simplified appreciably [**?**] :

$$\mathbf{A_I} \approx A^{lT} \begin{bmatrix} 1 & 0 & -a_1 \\ 0 & 1 & -a_2 \\ -a_1 & -a_2 & \varepsilon + a_1^2 + a_2^2 \end{bmatrix}$$

The resulting isopycnal operator conserves the quantity and dissipates its square. The demonstration of the first property is trivial as (B.2) is the divergence of fluxes. Let us demonstrate the second one :

$$\iiint_D T \, \nabla.\left(\mathbf{A_I} \nabla T\right) \, dv = - \iiint_D \nabla T \, .\left(\mathbf{A_I} \nabla T\right) \, dv$$

since

$$\begin{aligned} \nabla T \, .\left(\mathbf{A_I} \nabla T\right) &= A^{lT} \left[\left(\tfrac{\partial T}{\partial i}\right)^2 - 2a_1 \tfrac{\partial T}{\partial i}\tfrac{\partial T}{\partial k} + \left(\tfrac{\partial T}{\partial j}\right)^2 \right. \\ &\qquad \left. - 2a_2 \tfrac{\partial T}{\partial j}\tfrac{\partial T}{\partial k} + \left(a_1^2+a_2^2\right)\left(\tfrac{\partial T}{\partial k}\right)^2 \right] \\ &= A_h \left[\left(\tfrac{\partial T}{\partial i} - a_1 \tfrac{\partial T}{\partial k}\right)^2 + \left(\tfrac{\partial T}{\partial j} - a_2 \tfrac{\partial T}{\partial k}\right)^2 \right] \\ &\geq 0 \end{aligned}$$

the property becomes obvious.

The resulting diffusion operator in z-coordinate has the following form :

$$\begin{aligned} D^T = \frac{1}{e_1 e_2} \left\{ \frac{\partial}{\partial i}\left[A_h \left(\frac{e_2}{e_1}\frac{\partial T}{\partial i} - a_1 \frac{e_2}{e_3}\frac{\partial T}{\partial k} \right) \right] + \frac{\partial}{\partial j}\left[A_h \left(\frac{e_1}{e_2}\frac{\partial T}{\partial j} - a_2 \frac{e_1}{e_3}\frac{\partial T}{\partial k} \right) \right] \right\} \\ + \frac{1}{e_3}\frac{\partial}{\partial k}\left[A_h \left(-\frac{a_1}{e_1}\frac{\partial T}{\partial i} - \frac{a_2}{e_2}\frac{\partial T}{\partial j} + \frac{\left(a_1^2+a_2^2\right)}{e_3}\frac{\partial T}{\partial k} \right) \right] \end{aligned}$$

It has to be emphasised that the simplification introduced, leads to a decoupling between (i,z) and (j,z) planes. The operator has therefore the same expression as (**??**), the diffusion operator obtained for geopotential diffusion in the s-coordinate.

B.3 Lateral/Vertical Momentum Diffusive Operators

The second order momentum diffusion operator (Laplacian) in the z-coordinate is found by applying (2.11e), the expression for the Laplacian of a vector, to the horizontal velocity vector :

$$\Delta \mathbf{U}_h = \nabla \left(\nabla \cdot \mathbf{U}_h \right) - \nabla \times \left(\nabla \times \mathbf{U}_h \right)$$

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

$$= \begin{pmatrix} \frac{1}{e_1} \frac{\partial \chi}{\partial i} - \frac{1}{e_2} \frac{\partial \zeta}{\partial j} \\ \frac{1}{e_2} \frac{\partial \chi}{\partial j} + \frac{1}{e_1} \frac{\partial \zeta}{\partial i} \\ 0 \end{pmatrix} + \frac{1}{e_3} \begin{pmatrix} \frac{\partial}{\partial k} \left(\frac{1}{e_3} \frac{\partial u}{\partial k} \right) \\ \frac{\partial}{\partial k} \left(\frac{1}{e_3} \frac{\partial v}{\partial k} \right) \\ \frac{\partial \chi}{\partial k} - \frac{1}{e_1 e_2} \left(\frac{\partial^2 (e_2 \, u)}{\partial i \partial k} + \frac{\partial^2 (e_1 \, v)}{\partial j \partial k} \right) \end{pmatrix}$$

Using (2.11b), the definition of the horizontal divergence, the third componant of the second vector is obviously zero and thus :

$$\Delta \mathbf{U}_h = \nabla_h \left(\chi \right) - \nabla_h \times \left(\zeta \right) + \frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{1}{e_3} \frac{\partial \, \mathbf{U}_h}{\partial k} \right)$$

Note that this operator ensures a full separation between the vorticity and horizontal divergence fields (see Appendix E). It is only equal to a Laplacian applied to each component in Cartesian coordinates, not on the sphere.

The horizontal/vertical second order (Laplacian type) operator used to diffuse horizontal momentum in the z-coordinate therefore takes the following form :

$$\mathbf{D}^{\mathbf{U}} = \nabla_h \left(A^{lm} \, \chi \right) - \nabla_h \times \left(A^{lm} \, \zeta \, \mathbf{k} \right) + \frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \tag{B.6}$$

that is, in expanded form :

$$D_u^{\mathbf{U}} = \frac{1}{e_1} \frac{\partial \left(A^{lm} \chi \right)}{\partial i} - \frac{1}{e_2} \frac{\partial \left(A^{lm} \zeta \right)}{\partial j} + \frac{1}{e_3} \frac{\partial u}{\partial k}$$

$$D_v^{\mathbf{U}} = \frac{1}{e_2} \frac{\partial \left(A^{lm} \chi \right)}{\partial j} + \frac{1}{e_1} \frac{\partial \left(A^{lm} \zeta \right)}{\partial i} + \frac{1}{e_3} \frac{\partial v}{\partial k}$$

Note Bene : introducing a rotation in (B.6) does not lead to a useful expression for the iso/diapycnal Laplacian operator in the z-coordinate. Similarly, we did not found an expression of practical use for the geopotential horizontal/vertical Laplacian operator in the s-coordinate. Generally, (B.6) is used in both z- and s-coordinate systems, that is a Laplacian diffusion is applied on momentum along the coordinate directions.

C Discrete Invariants of the Equations

Contents

C.1 Introduction / Notations

Notation used in this appendix in the demonstations :
fluxes at the faces of a T-box :

$$U = e_{2u}\, e_{3u}\, u \qquad V = e_{1v}\, e_{3v}\, v \qquad W = e_{1w}\, e_{2w}\, \omega$$

volume of cells at u-, v-, and T-points :

$$b_u = e_{1u}\, e_{2u}\, e_{3u} \qquad b_v = e_{1v}\, e_{2v}\, e_{3v} \qquad b_t = e_{1t}\, e_{2t}\, e_{3t}$$

partial derivative notation : $\partial_\bullet = \frac{\partial}{\partial \bullet}$
$dv = e_1\, e_2\, e_3\, di\, dj\, dk$ is the volume element, with only e_3 that depends on time. D and S are the ocean domain volume and surface, respectively. No wetting/drying is allow ($i.e.$ $\frac{\partial S}{\partial t} = 0$) Let k_s and k_b be the ocean surface and bottom, resp. ($i.e.$ $s(k_s) = \eta$ and $s(k_b) = -H$, where H is the bottom depth).

$$z(k) = \eta - \int_{\tilde{k}=k}^{\tilde{k}=k_s} e_3(\tilde{k})\, d\tilde{k} = \eta - \int_{k}^{k_s} e_3\, d\tilde{k}$$

Continuity equation with the above notation :

$$\frac{1}{e_{3t}} \partial_t(e_{3t}) + \frac{1}{b_t} \left\{ \delta_i[U] + \delta_j[V] + \delta_k[W] \right\} = 0$$

A quantity, Q is conserved when its domain averaged time change is zero, that is when :

$$\partial_t \left(\int_D Q\, dv \right) = 0$$

Noting that the coordinate system used .... blah blah

$$\partial_t \left(\int_D Q\, dv \right) = \int_D \partial_t \left(e_3\, Q \right) e_1 e_2\, di\, dj\, dk = \int_D \frac{1}{e_3} \partial_t \left(e_3\, Q \right) dv = 0$$

equation of evolution of Q written as the time evolution of the vertical content of Q like for tracers, or momentum in flux form, the quadratic quantity $\frac{1}{2}Q^2$ is conserved when :

$$\begin{aligned} \partial_t \left(\int_D \frac{1}{2}\, Q^2\, dv \right) &= \int_D \frac{1}{2} \partial_t \left(\frac{1}{e_3} \left(e_3\, Q \right)^2 \right) e_1 e_2\, di\, dj\, dk \\ &= \int_D Q\, \partial_t \left(e_3\, Q \right) e_1 e_2\, di\, dj\, dk - \int_D \frac{1}{2} Q^2\, \partial_t(e_3)\, e_1 e_2\, di\, dj\, dk \end{aligned}$$

that is in a more compact form :

$$\partial_t \left(\int_D \frac{1}{2} Q^2 \, dv \right) = \int_D \frac{Q}{e_3} \partial_t \left(e_3 \, Q \right) dv - \frac{1}{2} \int_D \frac{Q^2}{e_3} \partial_t (e_3) \, dv \tag{C.1}$$

equation of evolution of Q written as the time evolution of Q like for momentum in vector invariant form, the quadratic quantity $\frac{1}{2}Q^2$ is conserved when :

$$\begin{aligned} \partial_t \left(\int_D \frac{1}{2} Q^2 \, dv \right) &= \int_D \frac{1}{2} \partial_t \left(e_3 \, Q^2 \right) \; e_1 e_2 \, di \, dj \, dk \\ &= \int_D Q \partial_t Q \; e_1 e_2 e_3 \, di \, dj \, dk + \int_D \frac{1}{2} Q^2 \, \partial_t e_3 \; e_1 e_2 \, di \, dj \, dk \end{aligned}$$

that is in a more compact form :

$$\partial_t \left(\int_D \frac{1}{2} Q^2 \, dv \right) = \int_D Q \, \partial_t Q \, dv + \frac{1}{2} \int_D \frac{1}{e_3} Q^2 \partial_t e_3 \, dv \tag{C.2}$$

C.2 Continuous conservation

The discretization of pimitive equation in s-coordinate ($i.e.$ time and space varying vertical coordinate) must be chosen so that the discrete equation of the model satisfy integral constrains on energy and enstrophy.

Let us first establish those constraint in the continuous world. The total energy ($i.e.$ kinetic plus potential energies) is conserved :

$$\partial_t \left(\int_D \left(\frac{1}{2} {\mathbf{U}_h}^2 + \rho \, g \, z \right) \; dv \right) = 0 \tag{C.3}$$

under the following assumptions : no dissipation, no forcing (wind, buoyancy flux, atmospheric pressure variations), mass conservation, and closed domain.

This equation can be transformed to obtain several sub-equalities. The transformation for the advection term depends on whether the vector invariant form or the flux form is used for the momentum equation. Using (C.2) and introducing (A.15) in (C.3) for the former form and Using (C.1) and introducing (A.16) in (C.3) for the latter form leads to :

advection term (vector invariant form) :

$$\int_D \zeta \; (\mathbf{k} \times \mathbf{U}_h) \cdot \mathbf{U}_h \, dv = 0 \tag{C.4a}$$

$$\int_D \mathbf{U}_h \cdot \nabla_h \left(\frac{{\mathbf{U}_h}^2}{2} \right) dv + \int_D \mathbf{U}_h \cdot \nabla_z \mathbf{U}_h \, dv - \int_D \frac{{\mathbf{U}_h}^2}{2} \frac{1}{e_3} \partial_t e_3 \, dv = 0 \tag{C.4b}$$

advection term (flux form) :

$$\int_D \frac{1}{e_1 e_2} \left(v\, \partial_i e_2 - u\, \partial_j e_1 \right) \; (\mathbf{k} \times \mathbf{U}_h) \cdot \mathbf{U}_h \; dv = 0 \tag{C.4c}$$

$$\int_D \mathbf{U}_h \cdot \begin{pmatrix} \nabla \cdot (\mathbf{U}\, u) \\ \nabla \cdot (\mathbf{U}\, v) \end{pmatrix} \; dv + \frac{1}{2} \int_D {\mathbf{U}_h}^2 \frac{1}{e_3} \partial_t e_3 \; dv = \; 0 \tag{C.4d}$$

coriolis term

$$\int_D f \; (\mathbf{k} \times \mathbf{U}_h) \cdot \mathbf{U}_h \; dv = 0 \tag{C.4e}$$

pressure gradient :

$$-\int_D \nabla p|_z \cdot \mathbf{U}_h \; dv = -\int_D \nabla \cdot (\rho\, \mathbf{U}) \; g\; z \; \; dv + \int_D g\, \rho\; \partial_t z \; dv \tag{C.4f}$$

where $\nabla_h = \nabla|_k$ is the gradient along the s-surfaces.
blah blah....
The prognostic ocean dynamics equation can be summarized as follows :

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

Vector invariant form :

$$\int_D \mathbf{U}_h \cdot \text{VOR} \; dv = 0 \tag{C.5a}$$

$$\int_D \mathbf{U}_h \cdot \text{KEG} \; dv + \int_D \mathbf{U}_h \cdot \text{ZAD} \; dv - \int_D \frac{{\mathbf{U}_h}^2}{2} \frac{1}{e_3} \partial_t e_3 \; dv = 0 \tag{C.5b}$$

$$-\int_D \mathbf{U}_h \cdot (\text{HPG} + \text{SPG}) \; dv = -\int_D \nabla \cdot (\rho\, \mathbf{U}) \; g\; z \; \; dv + \int_D g\, \rho\; \partial_t z \; dv \tag{C.5c}$$

Flux form :

$$\int_D \mathbf{U}_h \cdot \text{COR} \; dv = 0 \tag{C.6a}$$

$$\int_D \mathbf{U}_h \cdot \text{ADV} \; dv + \frac{1}{2} \int_D {\mathbf{U}_h}^2 \frac{1}{e_3} \partial_t e_3 \; dv = \; 0 \tag{C.6b}$$

$$-\int_D \mathbf{U}_h \cdot (\text{HPG} + \text{SPG}) \; dv = -\int_D \nabla \cdot (\rho\, \mathbf{U}) \; g\; z \; \; dv + \int_D g\, \rho\; \partial_t z \; dv \tag{C.6c}$$

(C.6c) is the balance between the conversion KE to PE and PE to KE. Indeed the left hand side of (C.6c) can be transformed as follows :

$$
\begin{aligned}
\partial_t \left(\int_D \rho\, g\, z\, dv \right) &= + \int_D \frac{1}{e_3} \partial_t (e_3\, \rho)\; g\; z \;\; dv + \int_D g\, \rho\, \partial_t z\, dv \\
&= - \int_D \nabla \cdot (\rho\, \mathbf{U}) \;\; g\; z \;\; dv + \int_D g\, \rho\, \partial_t z\, dv \\
&= + \int_D \rho\, g \left(\mathbf{U}_h \cdot \nabla_h z + \omega \frac{1}{e_3} \partial_k z \right) \; dv + \int_D g\, \rho\, \partial_t z\, dv \\
&= + \int_D \rho\, g\, (\omega + \partial_t z + \mathbf{U}_h \cdot \nabla_h z) \;\; dv \\
&= + \int_D g\, \rho\, w\; dv
\end{aligned}
$$

where the last equality is obtained by noting that the brackets is exactly the expression of w, the vertical velocity referenced to the fixe z-coordinate system (see (A.5)).

The left hand side of (C.6c) can be transformed as follows :

$$
\begin{aligned}
- \int_D \nabla p|_z \cdot \mathbf{U}_h \; dv &= - \int_D (\nabla_h p + \rho\, g \nabla_h z) \cdot \mathbf{U}_h \; dv \\
&= - \int_D \nabla_h p \cdot \mathbf{U}_h \; dv - \int_D \rho\, g\, \nabla_h z \cdot \mathbf{U}_h \; dv \\
&= + \int_D p\, \nabla_h \cdot \mathbf{U}_h \; dv + \int_D \rho\, g\, (\omega - w + \partial_t z) \;\; dv \\
&= - \int_D p \left(\frac{1}{e_3} \partial_t e_3 + \frac{1}{e_3} \partial_k \omega \right) \; dv + \int_D \rho\, g\, (\omega - w + \partial_t z) \;\; dv \\
&= - \int_D \frac{p}{e_3} \partial_t e_3 \; dv + \int_D \frac{1}{e_3} \partial_k p\; \omega \; dv + \int_D \rho\, g\, (\omega - w + \partial_t z) \;\; dv \\
&= - \int_D \frac{p}{e_3} \partial_t e_3 \; dv - \int_D \rho\, g\, \omega \; dv + \int_D \rho\, g\, (\omega - w + \partial_t z) \;\; dv \\
&= - \int_D \frac{p}{e_3} \partial_t e_3 \;\; dv - \int_D \rho\, g\, w \; dv + \int_D \rho\, g\, \partial_t z\; dv
\end{aligned}
$$

introducing the hydrostatic balance $\partial_k p = -\rho\, g\, e_3$ in the last term, it becomes :

$$
\begin{aligned}
&= -\int_D \frac{p}{e_3}\partial_t e_3 \; dv - \int_D \rho\, g\, w \; dv - \int_D \frac{1}{e_3}\partial_k p\, \partial_t z \; dv \\
&= -\int_D \frac{p}{e_3}\partial_t e_3 \; dv - \int_D \rho\, g\, w \; dv + \int_D \frac{p}{e_3}\partial_t(\partial_k z) dv \\
&= -\int_D \rho\, g\, w \; dv
\end{aligned}
$$

C.3 Discrete total energy conservation : vector invariant form

C.3.1 Total energy conservation

The discrete form of the total energy conservation, (C.3), is given by :

$$
\partial_t \left(\sum_{i,j,k} \left\{ \frac{u^2}{2}\, b_u + \frac{v^2}{2}\, b_v + \rho\, g\, z_t\, b_t \right\} \right) = 0
$$

which in vector invariant forms, it leads to :

$$
\begin{aligned}
\sum_{i,j,k} \left\{ u\, \partial_t u\; b_u + v\, \partial_t v\; b_v \right\} + \frac{1}{2} \sum_{i,j,k} \left\{ \frac{u^2}{e_{3u}} \partial_t e_{3u}\; b_u + \frac{v^2}{e_{3v}} \partial_t e_{3v}\; b_v \right\} \\
= -\sum_{i,j,k} \left\{ \frac{1}{e_{3t}} \partial_t (e_{3t}\rho)\, g\, z_t\; b_t \right\} - \sum_{i,j,k} \left\{ \rho\, g\, \partial_t(z_t)\, b_t \right\}
\end{aligned}
\tag{C.7}
$$

Substituting the discrete expression of the time derivative of the velocity either in vector invariant, leads to the discrete equivalent of the four equations (C.6).

C.3.2 Vorticity term (coriolis + vorticity part of the advection)

Let q, located at f-points, be either the relative ($q = \zeta / e_{3f}$), or the planetary ($q = f / e_{3f}$), or the total potential vorticity ($q = (\zeta + f) / e_{3f}$). Two discretisation of the vorticity term (ENE and EEN) allows the conservation of the kinetic energy.

Vorticity Term with ENE scheme (*ln_dynvor_ene*=.true.)

For the ENE scheme, the two components of the vorticity term are given by :

$$-e_3\, q\, \mathbf{k} \times \mathbf{U}_h \equiv \begin{pmatrix} +\frac{1}{e_{1u}}\ \overline{q\ \overline{(e_{1v}\, e_{3v}\, v)}^{\,i+1/2}}^{\,j} \\ -\frac{1}{e_{2v}}\ \overline{q\ \overline{(e_{2u}\, e_{3u}\, u)}^{\,j+1/2}}^{\,i} \end{pmatrix}$$

This formulation does not conserve the enstrophy but it does conserve the total kinetic energy. Indeed, the kinetic energy tendency associated to the vorticity term and averaged over the ocean domain can be transformed as follows :

$$\begin{aligned} &\int_D -\left(e_3\, q\, \mathbf{k} \times \mathbf{U}_h\right) \cdot \mathbf{U}_h\, dv \\ &\qquad \equiv \sum_{i,j,k} \left\{ \tfrac{1}{e_{1u}}\, \overline{q\, \overline{V}^{\,i+1/2}}^{\,j}\ u\, b_u - \tfrac{1}{e_{2v}}\, \overline{q\, \overline{U}^{\,j+1/2}}^{\,i}\ v\, b_v \right\} \\ &\qquad \equiv \sum_{i,j,k} \left\{ \overline{q\, \overline{V}^{\,i+1/2}}^{\,j}\ U - \overline{q\, \overline{U}^{\,j+1/2}}^{\,i}\ V \right\} \\ &\qquad \equiv \sum_{i,j,k} q \left\{ \overline{V}^{\,i+1/2}\, \overline{U}^{\,j+1/2} - \overline{U}^{\,j+1/2}\, \overline{V}^{\,i+1/2} \right\} \quad \equiv 0 \end{aligned}$$

In other words, the domain averaged kinetic energy does not change due to the vorticity term.

Vorticity Term with EEN scheme (*ln_dynvor_een*=.true.)

With the EEN scheme, the vorticity terms are represented as :

$$\begin{cases} +q\, e_3\, v \equiv +\dfrac{1}{e_{1u}} \displaystyle\sum_{i_p,\, k_p} {}^{\,i+1/2-i_p}_{\,j}\mathbb{Q}^{\,i_p}_{j_p}\, \left(e_{1v} e_{3v}\, v\right)^{\,i+i_p-1/2}_{\,j+j_p} \\ -q\, e_3\, u \equiv -\dfrac{1}{e_{2v}} \displaystyle\sum_{i_p,\, k_p} {}^{\,i}_{\,j+1/2-j_p}\mathbb{Q}^{\,i_p}_{j_p}\, \left(e_{2u} e_{3u}\, u\right)^{\,i+i_p}_{\,j+j_p-1/2} \end{cases} \tag{C.8}$$

where the indices i_p and k_p take the following value : $i_p = -1/2$ or $1/2$ and $j_p = -1/2$ or $1/2$, and the vorticity triads, ${}^{\,i}_{\,j}\mathbb{Q}^{\,i_p}_{j_p}$, defined at T-point, are given by :

$${}^{\,j}_{\,i}\mathbb{Q}^{\,i_p}_{j_p} = \frac{1}{12}\ \left(q^{\,i-i_p}_{j+j_p} + q^{\,i+j_p}_{j+i_p} + q^{\,i+i_p}_{j-j_p} \right) \tag{C.9}$$

This formulation does conserve the total kinetic energy. Indeed,

$$\int_D -\mathbf{U}_h \cdot (\zeta\, \mathbf{k} \times \mathbf{U}_h)\; dv$$

$$\equiv \sum_{i,j,k} \left\{ \left[\sum_{i_p,\, k_p} {}^{i+1/2-i_p}_{\;\;j}\mathbb{Q}^{i_p}_{j_p}\, V^{i+1/2-i_p}_{j+j_p} \right] U^{i+1/2}_j - \left[\sum_{i_p,\, k_p} {}^{\;\;i}_{j+1/2-j_p}\mathbb{Q}^{i_p}_{j_p}\, U^{i+i_p}_{j+1/2-j_p} \right] V^i_{j+1/2} \right\}$$

$$\equiv \sum_{i,j,k} \sum_{i_p,\, k_p} \left\{ \; {}^{i+1/2-i_p}_{\;\;j}\mathbb{Q}^{i_p}_{j_p}\, V^{i+1/2-i_p}_{j+j_p}\, U^{i+1/2}_j - {}^{\;\;i}_{j+1/2-j_p}\mathbb{Q}^{i_p}_{j_p}\, U^{i+i_p}_{j+1/2-j_p}\, V^i_{j+1/2} \; \right\}$$

Expending the summation on i_p and k_p, it becomes :

$$\begin{aligned}
\equiv \sum_{i,j,k} \Big\{ & \; {}^{i+1}_{j}\mathbb{Q}^{-1/2}_{+1/2}\, V^{i+1}_{j+1/2}\, U^{i+1/2}_j - {}^{\;\;i}_{j}\mathbb{Q}^{-1/2}_{+1/2}\, U^{i-1/2}_j\, V^i_{j+1/2} \\
& + {}^{i+1}_{j}\mathbb{Q}^{-1/2}_{-1/2}\, V^{i+1}_{j-1/2}\, U^{i+1/2}_j - {}^{\;\;i}_{j+1}\mathbb{Q}^{-1/2}_{-1/2}\, U^{i-1/2}_{j+1}\, V^i_{j+1/2} \\
& + {}^{i}_{j}\mathbb{Q}^{+1/2}_{+1/2}\, V^{i}_{j+1/2}\, U^{i+1/2}_j - {}^{\;\;i}_{j}\mathbb{Q}^{+1/2}_{+1/2}\, U^{i+1/2}_j\, V^i_{j+1/2} \\
& + {}^{i}_{j}\mathbb{Q}^{+1/2}_{-1/2}\, V^{i}_{j-1/2}\, U^{i+1/2}_j - {}^{\;\;i}_{j+1}\mathbb{Q}^{+1/2}_{-1/2}\, U^{i+1/2}_{j+1}\, V^i_{j+1/2} \; \Big\}
\end{aligned}$$

The summation is done over all i and j indices, it is therefore possible to introduce a shift of -1 either in i or j direction in some of the term of the summation (first term of the first and second lines, second term of the second and fourth lines). By doning so, we can regroup all the terms of the summation by triad at a (i,j) point. In other words, we regroup all the terms in the neighbourhood that contain a triad at the same (i,j) indices. It becomes :

$$\begin{aligned}
\equiv \sum_{i,j,k} \Big\{ & \; {}^{i}_{j}\mathbb{Q}^{-1/2}_{+1/2} \left[V^i_{j+1/2}\, U^{i-1/2}_j - U^{i-1/2}_j\, V^i_{j+1/2} \right] \\
& + {}^{i}_{j}\mathbb{Q}^{-1/2}_{-1/2} \left[V^i_{j-1/2}\, U^{i-1/2}_j - U^{i-1/2}_j\, V^i_{j-1/2} \right] \\
& + {}^{i}_{j}\mathbb{Q}^{+1/2}_{+1/2} \left[V^i_{j+1/2}\, U^{i+1/2}_j - U^{i+1/2}_j\, V^i_{j+1/2} \right] \\
& + {}^{i}_{j}\mathbb{Q}^{+1/2}_{-1/2} \left[V^i_{j-1/2}\, U^{i+1/2}_j - U^{i+1/2}_{j-1}\, V^i_{j-1/2} \right] \; \Big\} \qquad \equiv \; 0
\end{aligned}$$

Gradient of Kinetic Energy / Vertical Advection

The change of Kinetic Energy (KE) due to the vertical advection is exactly balanced by the change of KE due to the horizontal gradient of KE :

$$\int_D \mathbf{U}_h \cdot \frac{1}{e_3}\omega\partial_k \mathbf{U}_h \, dv = -\int_D \mathbf{U}_h \cdot \nabla_h \left(\frac{1}{2}\,{\mathbf{U}_h}^2 \right) \, dv + \frac{1}{2}\int_D \frac{{\mathbf{U}_h}^2}{e_3}\partial_t(e_3) \, dv$$

Indeed, using successively (4.11) (*i.e.* the skew symmetry property of the δ operator) and the continuity equation, then (4.11) again, then the commutativity of operators $\overline{\cdot}$ and δ, and finally (4.12) (*i.e.* the symmetry property of the $\overline{\cdot}$ operator) applied in the horizontal and vertical directions, it becomes :

$$\begin{aligned}
&-\int_D \mathbf{U}_h \cdot \text{KEG} \, dv = -\int_D \mathbf{U}_h \cdot \nabla_h \left(\frac{1}{2}\,{\mathbf{U}_h}^2 \right) \, dv \\
&\equiv -\sum_{i,j,k} \frac{1}{2}\left\{ \frac{1}{e_{1u}}\delta_{i+1/2}\left[\overline{u^2}^{\,i} + \overline{v^2}^{\,j} \right] u\, b_u + \frac{1}{e_{2v}}\delta_{j+1/2}\left[\overline{u^2}^{\,i} + \overline{v^2}^{\,j} \right] v\, b_v \right\} \\
&\equiv +\sum_{i,j,k} \frac{1}{2}\left(\overline{u^2}^{\,i} + \overline{v^2}^{\,j} \right) \left\{ \delta_i\left[U\right] + \delta_j\left[V\right] \right\} \\
&\equiv -\sum_{i,j,k} \frac{1}{2}\left(\overline{u^2}^{\,i} + \overline{v^2}^{\,j} \right) \left\{ \frac{b_t}{e_{3t}}\partial_t(e_{3t}) + \delta_k\left[W\right] \right\} \\
&\equiv +\sum_{i,j,k} \frac{1}{2}\delta_{k+1/2}\left[\overline{u^2}^{\,i} + \overline{v^2}^{\,j} \right] W - \sum_{i,j,k} \frac{1}{2}\left(\overline{u^2}^{\,i} + \overline{v^2}^{\,j} \right) \partial_t b_t \\
&\equiv +\sum_{i,j,k} \frac{1}{2}\left(\overline{\delta_{k+1/2}\left[u^2\right]}^{\,i} + \overline{\delta_{k+1/2}\left[v^2\right]}^{\,j} \right) W - \sum_{i,j,k} \left(\frac{u^2}{2}\,\partial_t \overline{b_t}^{\,i+1/2} + \frac{v^2}{2}\,\partial_t \overline{b_t}^{\,j+1/2} \right)
\end{aligned}$$

Assuming that $b_u = \overline{b_t}^{\,i+1/2}$ and $b_v = \overline{b_t}^{\,j+1/2}$, or at least that the time derivative of these two equations is satisfied, it becomes :

$$\begin{aligned}
&\equiv \sum_{i,j,k} \frac{1}{2}\left\{ \overline{W}^{\,i+1/2}\, \delta_{k+1/2}\left[u^2\right] + \overline{W}^{\,j+1/2}\, \delta_{k+1/2}\left[v^2\right] \right\} - \sum_{i,j,k} \left(\frac{u^2}{2}\,\partial_t b_u + \frac{v^2}{2}\,\partial_t b_v \right) \\
&\equiv \sum_{i,j,k} \left\{ \overline{W}^{\,i+1/2}\, \overline{u}^{\,k+1/2}\, \delta_{k+1/2}[u] + \overline{W}^{\,j+1/2}\, \overline{v}^{\,k+1/2}\, \delta_{k+1/2}[v] \right\} - \sum_{i,j,k} \left(\frac{u^2}{2}\,\partial_t b_u + \frac{v^2}{2}\,\partial_t b_v \right) \\
&\equiv \sum_{i,j,k} \left\{ \frac{1}{b_u}\,\overline{\overline{W}^{\,i+1/2}\,\delta_{k+1/2}\left[u\right]}^{\,k}\, u\, b_u + \frac{1}{b_v}\,\overline{\overline{W}^{\,j+1/2}\delta_{k+1/2}\left[v\right]}^{\,k}\, v\, b_v \right\} - \sum_{i,j,k} \left(\frac{u^2}{2}\,\partial_t b_u + \frac{v^2}{2}\,\partial_t b_v \right)
\end{aligned}$$

The first term provides the discrete expression for the vertical advection of momentum (ZAD), while the second term corresponds exactly to (C.7), therefore :

$$\equiv \int_D \mathbf{U}_h \cdot \text{ZAD}\ dv + \frac{1}{2} \int_D {\mathbf{U}_h}^2 \frac{1}{e_3} \partial_t(e_3)\ dv$$
$$\equiv \int_D \mathbf{U}_h \cdot w \partial_k \mathbf{U}_h\ dv + \frac{1}{2} \int_D {\mathbf{U}_h}^2 \frac{1}{e_3} \partial_t(e_3)\ dv$$

There is two main points here. First, the satisfaction of this property links the choice of the discrete formulation of the vertical advection and of the horizontal gradient of KE. Choosing one imposes the other. For example KE can also be discretized as $1/2\,(\overline{u}^{\,i^2} + \overline{v}^{\,j^2})$. This leads to the following expression for the vertical advection :

$$\frac{1}{e_3}\ \omega\ \partial_k \mathbf{U}_h \equiv \begin{pmatrix} \frac{1}{e_{1u}\, e_{2u}\, e_{3u}}\ \overline{\overline{e_{1t}\, e_{2t}\, \omega\ \delta_{k+1/2}\left[\overline{u}^{\,i+1/2}\right]}}^{\,i+1/2,k} \\ \frac{1}{e_{1v}\, e_{2v}\, e_{3v}}\ \overline{\overline{e_{1t}\, e_{2t}\, \omega\ \delta_{k+1/2}\left[\overline{v}^{\,j+1/2}\right]}}^{\,j+1/2,k} \end{pmatrix}$$

a formulation that requires an additional horizontal mean in contrast with the one used in NEMO. Nine velocity points have to be used instead of 3. This is the reason why it has not been chosen.

Second, as soon as the chosen s-coordinate depends on time, an extra constraint arises on the time derivative of the volume at u- and v-points :

$$e_{1u}\, e_{2u}\, \partial_t(e_{3u}) = \overline{e_{1t}\, e_{2t}\, \partial_t(e_{3t})}^{\,i+1/2}$$
$$e_{1v}\, e_{2v}\, \partial_t(e_{3v}) = \overline{e_{1t}\, e_{2t}\, \partial_t(e_{3t})}^{\,j+1/2}$$

which is (over-)satified by defining the vertical scale factor as follows :

$$e_{3u} = \frac{1}{e_{1u}\, e_{2u}}\ \overline{e_{1t}\, e_{2t}\, e_{3t}}^{\,i+1/2} \tag{C.10}$$
$$e_{3v} = \frac{1}{e_{1v}\, e_{2v}}\ \overline{e_{1t}\, e_{2t}\, e_{3t}}^{\,j+1/2} \tag{C.11}$$

Blah blah required on the the step representation of bottom topography.....

C.3.3 Pressure Gradient Term

When the equation of state is linear ($i.e.$ when an advection-diffusion equation for density can be derived from those of temperature and salinity) the change of KE due to the work of pressure forces is balanced by the change of potential energy due to buoyancy forces :

$$-\int_D \nabla p|_z \cdot \mathbf{U}_h\ dv = -\int_D \nabla \cdot (\rho\, \mathbf{U})\ g\, z\ dv + \int_D g\, \rho\ \partial_t(z)\ dv$$

This property can be satisfied in a discrete sense for both z- and s-coordinates. Indeed, defining the depth of a T-point, z_t, as the sum of the vertical scale factors at w-points starting from the surface, the work of pressure forces can be written as :

$$
\begin{aligned}
-\int_D \nabla p|_z \cdot \mathbf{U}_h \, dv \equiv \sum_{i,j,k} \Bigg\{ & -\frac{1}{e_{1u}} \Big(\delta_{i+1/2}[p_t] - g \, \overline{\rho}^{\,i+1/2} \, \delta_{i+1/2}[z_t] \Big) \, u \, b_u \\
& -\frac{1}{e_{2v}} \Big(\delta_{j+1/2}[p_t] - g \, \overline{\rho}^{\,j+1/2} \delta_{j+1/2}[z_t] \Big) \, v \, b_v \Bigg\}
\end{aligned}
$$

Using successively (4.11), *i.e.* the skew symmetry property of the δ operator, (6.4), the continuity equation, (6.20), the hydrostatic equation in the s-coordinate, and $\delta_{k+1/2}\left[z_t\right] \equiv e_{3w}$, which comes from the definition of z_t, it becomes :

$$
\begin{aligned}
&\equiv + \sum_{i,j,k} g \Big\{ \overline{\rho}^{\,i+1/2} \, U \, \delta_{i+1/2}[z_t] + \overline{\rho}^{\,j+1/2} \, V \, \delta_{j+1/2}[z_t] + \Big(\delta_i[U] + \delta_j[V] \Big) \, \frac{p_t}{g} \Big\} \\
&\equiv + \sum_{i,j,k} g \Big\{ \overline{\rho}^{\,i+1/2} \, U \, \delta_{i+1/2}[z_t] + \overline{\rho}^{\,j+1/2} \, V \, \delta_{j+1/2}[z_t] - \Big(\frac{b_t}{e_{3t}} \partial_t(e_{3t}) + \delta_k \left[W \right] \Big) \, \frac{p_t}{g} \Big\} \\
&\equiv + \sum_{i,j,k} g \Big\{ \overline{\rho}^{\,i+1/2} \, U \, \delta_{i+1/2}[z_t] + \overline{\rho}^{\,j+1/2} \, V \, \delta_{j+1/2}[z_t] + \frac{W}{g} \, \delta_{k+1/2}[p_t] - \frac{p_t}{g} \, \partial_t b_t \Big\} \\
&\equiv + \sum_{i,j,k} g \Big\{ \overline{\rho}^{\,i+1/2} \, U \, \delta_{i+1/2}[z_t] + \overline{\rho}^{\,j+1/2} \, V \, \delta_{j+1/2}[z_t] - W \, e_{3w} \overline{\rho}^{\,k+1/2} - \frac{p_t}{g} \, \partial_t b_t \Big\} \\
&\equiv + \sum_{i,j,k} g \Big\{ \overline{\rho}^{\,i+1/2} \, U \, \delta_{i+1/2}[z_t] + \overline{\rho}^{\,j+1/2} \, V \, \delta_{j+1/2}[z_t] + W \, \overline{\rho}^{\,k+1/2} \, \delta_{k+1/2}[z_t] - \frac{p_t}{g} \, \partial_t b_t \Big\} \\
&\equiv - \sum_{i,j,k} g \, z_t \Big\{ \delta_i \Big[U \, \overline{\rho}^{\,i+1/2} \Big] + \delta_j \Big[V \, \overline{\rho}^{\,j+1/2} \Big] + \delta_k \Big[W \, \overline{\rho}^{\,k+1/2} \Big] \Big\} - \sum_{i,j,k} \Big\{ p_t \, \partial_t b_t \Big\} \\
&\equiv + \sum_{i,j,k} g \, z_t \Big\{ \partial_t(e_{3t} \, \rho) \Big\} \, b_t - \sum_{i,j,k} \Big\{ p_t \, \partial_t b_t \Big\}
\end{aligned}
$$

The first term is exactly the first term of the right-hand-side of (C.7). It remains to demonstrate that the last term, which is obviously a discrete analogue of $\int_D \frac{p}{e_3} \partial_t(e_3) \, dv$ is equal to the last term of (C.7). In other words, the following property must be satisfied :

$$
\sum_{i,j,k} \Big\{ p_t \, \partial_t b_t \Big\} \equiv \sum_{i,j,k} \Big\{ \rho \, g \, \partial_t(z_t) \, b_t \Big\}
$$

Let introduce p_w the pressure at w-point such that $\delta_k[p_w] = -\rho \, g \, e_{3t}$. The right-hand-side of the above equation can be transformed as follows :

$$\sum_{i,j,k}\left\{\rho\, g\, \partial_t(z_t)\, b_t\right\} \equiv -\sum_{i,j,k}\left\{\delta_k[p_w]\, \partial_t(z_t)\, e_{1t}\, e_{2t}\right\}$$
$$\equiv +\sum_{i,j,k}\left\{p_w\, \delta_{k+1/2}[\partial_t(z_t)]\, e_{1t}\, e_{2t}\right\} \equiv +\sum_{i,j,k}\left\{p_w\, \partial_t(e_{3w})\, e_{1t}\, e_{2t}\right\}$$
$$\equiv +\sum_{i,j,k}\left\{p_w\, \partial_t(b_w)\right\}$$

therefore, the balance to be satisfied is :

$$\sum_{i,j,k}\left\{p_t\, \partial_t(b_t)\right\} \equiv \sum_{i,j,k}\left\{p_w\, \partial_t(b_w)\right\}$$

which is a purely vertical balance :

$$\sum_{k}\left\{p_t\, \partial_t(e_{3t})\right\} \equiv \sum_{k}\left\{p_w\, \partial_t(e_{3w})\right\}$$

Defining $p_w = \overline{p_t}^{\,k+1/2}$

Note that this property strongly constrains the discrete expression of both the depth of $T-$points and of the term added to the pressure gradient in the s-coordinate. Nevertheless, it is almost never satisfied since a linear equation of state is rarely used.

C.4 Discrete total energy conservation : flux form

C.4.1 Total energy conservation

The discrete form of the total energy conservation, (C.3), is given by :

$$\partial_t\left(\sum_{i,j,k}\left\{\frac{u^2}{2}\, b_u + \frac{v^2}{2}\, b_v + \rho\, g\, z_t\, b_t\right\}\right) = 0$$

which in flux form, it leads to :

$$\sum_{i,j,k}\left\{\frac{u}{e_{3u}}\frac{\partial(e_{3u}u)}{\partial t}\, b_u + \frac{v}{e_{3v}}\frac{\partial(e_{3v}v)}{\partial t}\, b_v\right\} - \frac{1}{2}\sum_{i,j,k}\left\{\frac{u^2}{e_{3u}}\frac{\partial e_{3u}}{\partial t}\, b_u + \frac{v^2}{e_{3v}}\frac{\partial e_{3v}}{\partial t}\, b_v\right\}$$
$$= -\sum_{i,j,k}\left\{\frac{1}{e_3 t}\frac{\partial e_{3t}\rho}{\partial t}\, g\, z_t\, b_t\right\} - \sum_{i,j,k}\left\{\rho\, g\, \frac{\partial z_t}{\partial t}\, b_t\right\}$$

Substituting the discrete expression of the time derivative of the velocity either in vector invariant or in flux form, leads to the discrete equivalent of the

C.4.2 Coriolis and advection terms : flux form

Coriolis plus "metric" Term

In flux from the vorticity term reduces to a Coriolis term in which the Coriolis parameter has been modified to account for the "metric" term. This altered Coriolis parameter is discretised at an f-point. It is given by :

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

Either the ENE or EEN scheme is then applied to obtain the vorticity term in flux form. It therefore conserves the total KE. The derivation is the same as for the vorticity term in the vector invariant form (§C.3.2).

Flux form advection

The flux form operator of the momentum advection is evaluated using a centered second order finite difference scheme. Because of the flux form, the discrete operator does not contribute to the global budget of linear momentum. Because of the centered second order scheme, it conserves the horizontal kinetic energy, that is :

$$-\int_D \mathbf{U}_h \cdot \begin{pmatrix} \nabla\cdot(\mathbf{U}\,u) \\ \nabla\cdot(\mathbf{U}\,v) \end{pmatrix} dv - \frac{1}{2}\int_D {\mathbf{U}_h}^2 \frac{1}{e_3}\frac{\partial e_3}{\partial t}\, dv = 0 \tag{C.12}$$

Let us first consider the first term of the scalar product ($i.e.$ just the the terms associated with the i-component of the advection) :

$$\begin{aligned}
&-\int_D u\cdot\nabla\cdot(\mathbf{U}\,u)\; dv \\
&\equiv -\sum_{i,j,k}\left\{ \frac{1}{b_u}\left(\delta_{i+1/2}\left[\overline{U}^{\,i}\,\overline{u}^{\,i}\right] + \delta_j\left[\overline{V}^{\,i+1/2}\,\overline{u}^{\,j+1/2}\right] + \delta_k\left[\overline{W}^{\,i+1/2}\,\overline{u}^{\,k+1/2}\right]\right)\right\} b_u\, u \\
&\equiv -\sum_{i,j,k}\left\{ \delta_{i+1/2}\left[\overline{U}^{\,i}\,\overline{u}^{\,i}\right] + \delta_j\left[\overline{V}^{\,i+1/2}\,\overline{u}^{\,j+1/2}\right] + \delta_k\left[\overline{W}^{\,i+12}\,\overline{u}^{\,k+1/2}\right]\right\} u \\
&\equiv +\sum_{i,j,k}\left\{ \overline{U}^{\,i}\,\overline{u}^{\,i}\delta_i\left[u\right] + \overline{V}^{\,i+1/2}\,\overline{u}^{\,j+1/2}\delta_{j+1/2}\left[u\right] + \overline{W}^{\,i+1/2}\,\overline{u}^{\,k+1/2}\delta_{k+1/2}\left[u\right]\right\} \\
&\equiv +\frac{1}{2}\sum_{i,j,k}\left\{ \overline{U}^{\,i}\delta_i\left[u^2\right] + \overline{V}^{\,i+1/2}\delta_{j+/2}\left[u^2\right] + \overline{W}^{\,i+1/2}\delta_{k+1/2}\left[u^2\right]\right\} \\
&\equiv -\sum_{i,j,k}\frac{1}{2}\left\{ U\;\delta_{i+1/2}\left[\overline{u^2}^{\,i}\right] + V\;\delta_{j+1/2}\left[\overline{u^2}^{\,i}\right] + W\;\delta_{k+1/2}\left[\overline{u^2}^{\,i}\right]\right\} \\
&\equiv -\sum_{i,j,k}\frac{1}{2}\overline{u^2}^{\,i}\left\{ \delta_{i+1/2}\left[U\right] + \delta_{j+1/2}\left[V\right] + \delta_{k+1/2}\left[W\right]\right\}
\end{aligned}$$

$$\equiv + \sum_{i,j,k} \frac{1}{2} \overline{u^2}^{\,i} \left\{ \left(\frac{1}{e_{3t}} \frac{\partial e_{3t}}{\partial t} \right) \; b_t \right\}$$

Applying similar manipulation applied to the second term of the scalar product leads to :

$$- \int_D \mathbf{U}_h \cdot \begin{pmatrix} \nabla \cdot (\mathbf{U}\, u) \\ \nabla \cdot (\mathbf{U}\, v) \end{pmatrix} \; dv \equiv + \sum_{i,j,k} \frac{1}{2} \left(\overline{u^2}^{\,i} + \overline{v^2}^{\,j} \right) \left\{ \left(\frac{1}{e_{3t}} \frac{\partial e_{3t}}{\partial t} \right) \; b_t \right\}$$

which is the discrete form of $\frac{1}{2} \int_D u \cdot \nabla \cdot (\mathbf{U}\, u) \; dv$. (C.12) is thus satisfied.

When the UBS scheme is used to evaluate the flux form momentum advection, the discrete operator does not contribute to the global budget of linear momentum (flux form). The horizontal kinetic energy is not conserved, but forced to decay ($i.e.$ the scheme is diffusive).

C.5 Discrete enstrophy conservation

Vorticity Term with ENS scheme (*ln_dynvor_ens*=.true.)

In the ENS scheme, the vorticity term is descretized as follows :

$$\begin{cases} + \dfrac{1}{e_{1u}} \overline{q}^{\,i} & \overline{\overline{(e_{1v}\, e_{3v}\, v)}}^{\,i,j+1/2} \\ - \dfrac{1}{e_{2v}} \overline{q}^{\,j} & \overline{\overline{(e_{2u}\, e_{3u}\, u)}}^{\,i+1/2,j} \end{cases} \tag{C.13}$$

The scheme does not allow but the conservation of the total kinetic energy but the conservation of q^2, the potential enstrophy for a horizontally non-divergent flow ($i.e.$ when χ=0). Indeed, using the symmetry or skew symmetry properties of the operators (Eqs (4.12) and (4.11)), it can be shown that :

$$\int_D q \;\; \mathbf{k} \cdot \frac{1}{e_3} \nabla \times (e_3\, q\, \mathbf{k} \times \mathbf{U}_h) \; dv \equiv 0 \tag{C.14}$$

where $dv = e_1\, e_2\, e_3 \; di\, dj\, dk$ is the volume element. Indeed, using (C.13), the discrete form of the right hand side of (C.14) can be transformed as follow :

$$\begin{aligned} &\int_D q \;\; \mathbf{k} \cdot \frac{1}{e_3} \nabla \times (e_3\, q\, \mathbf{k} \times \mathbf{U}_h) \; dv \\ &\qquad \equiv \sum_{i,j,k} q \; \left\{ \delta_{i+1/2} \left[-\overline{q}^{\,i} \, \overline{\overline{U}}^{\,i,j+1/2} \right] - \delta_{j+1/2} \left[\overline{q}^{\,j} \, \overline{\overline{V}}^{\,i+1/2,j} \right] \right\} \\ &\qquad \equiv \sum_{i,j,k} \left\{ \delta_i[q] \; \overline{q}^{\,i} \, \overline{\overline{U}}^{\,i,j+1/2} + \delta_j[q] \; \overline{q}^{\,j} \, \overline{\overline{V}}^{\,i+1/2,j} \right\} \\ &\qquad \equiv \tfrac{1}{2} \sum_{i,j,k} \left\{ \delta_i \left[q^2 \right] \; \overline{\overline{U}}^{\,i,j+1/2} + \delta_j \left[q^2 \right] \; \overline{\overline{V}}^{\,i+1/2,j} \right\} \\ &\qquad \equiv -\tfrac{1}{2} \sum_{i,j,k} q^2 \; \left\{ \delta_{i+1/2} \left[\overline{\overline{U}}^{\,i,j+1/2} \right] + \delta_{j+1/2} \left[\overline{\overline{V}}^{\,i+1/2,j} \right] \right\} \end{aligned}$$

Since $\overline{\cdot}$ and δ operators commute : $\delta_{i+1/2}\left[\overline{a}^{\,i}\right] = \overline{\delta_i\left[a\right]}^{\,i+1/2}$, and introducing the horizontal divergence χ, it becomes :

$$\equiv \sum_{i,j,k} -\tfrac{1}{2} q^2 \, \overline{\overline{e_{1t}\, e_{2t}\, e_{3t}\, \chi}}^{\,i+1/2,j+1/2} \qquad \equiv 0$$

The later equality is obtain only when the flow is horizontally non-divergent, $i.e.$ χ=0.

Vorticity Term with EEN scheme (*ln_dynvor_een*=.true.)

With the EEN scheme, the vorticity terms are represented as :

$$\begin{cases} +q\, e_3\, v \equiv +\dfrac{1}{e_{1u}} \displaystyle\sum_{i_p,\,k_p} {}^{\,i+1/2-i_p}_{\;\;j}\mathbb{Q}^{i_p}_{j_p}\, \left(e_{1v} e_{3v}\, v\right)^{i+i_p-1/2}_{j+j_p} \\ -q\, e_3\, u \equiv -\dfrac{1}{e_{2v}} \displaystyle\sum_{i_p,\,k_p} {}^{\;\;i}_{j+1/2-j_p}\mathbb{Q}^{i_p}_{j_p}\, \left(e_{2u} e_{3u}\, u\right)^{i+i_p}_{j+j_p-1/2} \end{cases} \tag{C.15}$$

where the indices i_p and k_p take the following value : $i_p = -1/2$ or $1/2$ and $j_p = -1/2$ or $1/2$, and the vorticity triads, ${}^{\,i}_{j}\mathbb{Q}^{i_p}_{j_p}$, defined at T-point, are given by :

$${}^{\,j}_{i}\mathbb{Q}^{i_p}_{j_p} = \frac{1}{12}\, \left(q^{\,i-i_p}_{j+j_p} + q^{\,i+j_p}_{j+i_p} + q^{\,i+i_p}_{j-j_p} \right) \tag{C.16}$$

This formulation does conserve the potential enstrophy for a horizontally non-divergent flow ($i.e.$ $\chi = 0$).

Let consider one of the vorticity triad, for example ${}^{\,i}_{j}\mathbb{Q}^{+1/2}_{+1/2}$, similar manipulation can be done for the 3 others. The discrete form of the right hand side of (C.14) applied to this triad only can be transformed as follow :

$$\begin{aligned}
&\int_D q\ \mathbf{k}\cdot\frac{1}{e_3}\nabla\times\left(e_3\, q\, \mathbf{k}\times\mathbf{U}_h\right)\ dv \\
\equiv &\sum_{i,j,k} q\, \left\{\ \delta_{i+1/2}\left[-\,{}^{\,i}_{j}\mathbb{Q}^{+1/2}_{+1/2}\, U^{i+1/2}_{j}\right] - \delta_{j+1/2}\left[{}^{\,i}_{j}\mathbb{Q}^{+1/2}_{+1/2}\, V^{i}_{j+1/2}\right]\ \right\} \\
\equiv &\sum_{i,j,k}\left\{\delta_i[q]\ {}^{\,i}_{j}\mathbb{Q}^{+1/2}_{+1/2}\, U^{i+1/2}_{j} + \delta_j[q]\ {}^{\,i}_{j}\mathbb{Q}^{+1/2}_{+1/2}\, V^{i}_{j+1/2}\right\} \\
&\ldots \\
&\textit{Demonstation to be done...} \\
&\ldots \\
\equiv &\frac{1}{2}\sum_{i,j,k}\left\{\delta_i\left[\left({}^{\,i}_{j}\mathbb{Q}^{+1/2}_{+1/2}\right)^2\right]\overline{\overline{U}}^{\,i,j+1/2} + \delta_j\left[\left({}^{\,i}_{j}\mathbb{Q}^{+1/2}_{+1/2}\right)^2\right]\overline{\overline{V}}^{\,i+1/2,j}\right\} \\
\equiv &-\frac{1}{2}\sum_{i,j,k}\left({}^{\,i}_{j}\mathbb{Q}^{+1/2}_{+1/2}\right)^2\ \left\{\delta_{i+1/2}\left[\overline{\overline{U}}^{\,i,j+1/2}\right] + \delta_{j+1/2}\left[\overline{\overline{V}}^{\,i+1/2,j}\right]\right\}
\end{aligned}$$

$$\equiv \sum_{i,j,k} -\frac{1}{2} \left({}^{i}_{j}\mathbb{Q}^{+1/2}_{+1/2} \right)^2 \overline{\overline{b_t}}^{\,i+1/2,\,j+1/2} \chi$$
$$\equiv 0$$

C.6 Conservation Properties on Tracers

All the numerical schemes used in NEMO are written such that the tracer content is conserved by the internal dynamics and physics (equations in flux form). For advection, only the CEN2 scheme ($i.e.$ 2^{nd} order finite different scheme) conserves the global variance of tracer. Nevertheless the other schemes ensure that the global variance decreases ($i.e.$ they are at least slightly diffusive). For diffusion, all the schemes ensure the decrease of the total tracer variance, except the iso-neutral operator. There is generally no strict conservation of mass, as the equation of state is non linear with respect to T and S. In practice, the mass is conserved to a very high accuracy.

C.6.1 Advection Term

conservation of a tracer, T :

$$\frac{\partial}{\partial t} \left(\int_D T \, dv \right) = \int_D \frac{1}{e_3} \frac{\partial \left(e_3 \, T \right)}{\partial t} \, dv = 0$$

conservation of its variance :

$$\frac{\partial}{\partial t} \left(\int_D \frac{1}{2} T^2 \, dv \right) = \int_D \frac{1}{e_3} Q \frac{\partial \left(e_3 \, T \right)}{\partial t} \, dv - \frac{1}{2} \int_D T^2 \frac{1}{e_3} \frac{\partial e_3}{\partial t} \, dv$$

Whatever the advection scheme considered it conserves of the tracer content as all the scheme are written in flux form. Indeed, let T be the tracer and τ_u, τ_v, and τ_w its interpolated values at velocity point (whatever the interpolation is), the conservation of the tracer content due to the advection tendency is obtained as follows :

$$\int_D \frac{1}{e_3} \frac{\partial \left(e_3 \, T \right)}{\partial t} \, dv = - \int_D \nabla \cdot \left(T \mathbf{U} \right) \, dv$$
$$\equiv - \sum_{i,j,k} \left\{ \frac{1}{b_t} \left(\delta_i \left[U \, \tau_u \right] + \delta_j \left[V \, \tau_v \right] \right) + \frac{1}{e_{3t}} \delta_k \left[w \, \tau_w \right] \right\} b_t$$
$$\equiv - \sum_{i,j,k} \left\{ \delta_i \left[U \, \tau_u \right] + \delta_j \left[V \, \tau_v \right] + \delta_k \left[W \, \tau_w \right] \right\}$$
$$\equiv 0$$

The conservation of the variance of tracer due to the advection tendency can be achieved only with the CEN2 scheme, $i.e.$ when $\tau_u = \overline{T}^{\,i+1/2}$, $\tau_v = \overline{T}^{\,j+1/2}$, and

$\tau_w = \overline{T}^{k+1/2}$. It can be demonstarted as follows :

$$
\begin{aligned}
&\int_D \frac{1}{e_3} Q \frac{\partial (e_3\, T)}{\partial t}\, dv = -\int_D \tau\, \nabla \cdot (T\, \mathbf{U})\, dv \\
&\equiv -\sum_{i,j,k} T\, \left\{ \delta_i \left[U \overline{T}^{i+1/2} \right] + \delta_j \left[V \overline{T}^{j+1/2} \right] + \delta_k \left[W \overline{T}^{k+1/2} \right] \right\} \\
&\equiv +\sum_{i,j,k} \left\{ U \overline{T}^{i+1/2}\, \delta_{i+1/2}\left[T\right] + V \overline{T}^{j+1/2}\, \delta_{j+1/2}\left[T\right] + W \overline{T}^{k+1/2}\, \delta_{k+1/2}\left[T\right] \right\} \\
&\equiv +\frac{1}{2} \sum_{i,j,k} \left\{ U\, \delta_{i+1/2}\left[T^2\right] + V\, \delta_{j+1/2}\left[T^2\right] + W\, \delta_{k+1/2}\left[T^2\right] \right\} \\
&\equiv -\frac{1}{2} \sum_{i,j,k} T^2 \left\{ \delta_i\left[U\right] + \delta_j\left[V\right] + \delta_k\left[W\right] \right\} \\
&\equiv +\frac{1}{2} \sum_{i,j,k} T^2 \left\{ \frac{1}{e_{3t}} \frac{\partial e_{3t}\, T}{\partial t} \right\}
\end{aligned}
$$

which is the discrete form of $\frac{1}{2} \int_D T^2 \frac{1}{e_3} \frac{\partial e_3}{\partial t}\, dv$.

C.7 Conservation Properties on Lateral Momentum Physics

The discrete formulation of the horizontal diffusion of momentum ensures the conservation of potential vorticity and the horizontal divergence, and the dissipation of the square of these quantities (i.e. enstrophy and the variance of the horizontal divergence) as well as the dissipation of the horizontal kinetic energy. In particular, when the eddy coefficients are horizontally uniform, it ensures a complete separation of vorticity and horizontal divergence fields, so that diffusion (dissipation) of vorticity (enstrophy) does not generate horizontal divergence (variance of the horizontal divergence) and *vice versa*.

These properties of the horizontal diffusion operator are a direct consequence of properties (4.9) and (4.10). When the vertical curl of the horizontal diffusion of momentum (discrete sense) is taken, the term associated with the horizontal gradient of the divergence is locally zero.

C.7.1 Conservation of Potential Vorticity

The lateral momentum diffusion term conserves the potential vorticity :

$$
\int_D \frac{1}{e_3} \mathbf{k} \cdot \nabla \times \left[\nabla_h \left(A^{lm}\, \chi \right) - \nabla_h \times \left(A^{lm}\, \zeta\, \mathbf{k} \right) \right] dv = 0
$$

$$= \int_D -\frac{1}{e_3}\mathbf{k}\cdot\nabla\times\left[\nabla_h\times\left(A^{lm}\,\zeta\,\mathbf{k}\right)\right]\,dv$$

$$\equiv \sum_{i,j}\left\{\delta_{i+1/2}\left[\frac{e_{2v}}{e_{1v}\,e_{3v}}\delta_i\left[A_f^{lm}e_{3f}\zeta\right]\right]+\delta_{j+1/2}\left[\frac{e_{1u}}{e_{2u}\,e_{3u}}\delta_j\left[A_f^{lm}e_{3f}\zeta\right]\right]\right\}$$

Using (4.11), it follows :

$$\equiv \sum_{i,j,k} -\left\{\frac{e_{2v}}{e_{1v}\,e_{3v}}\delta_i\left[A_f^{lm}e_{3f}\zeta\right]\;\delta_i\left[1\right]+\frac{e_{1u}}{e_{2u}\,e_{3u}}\delta_j\left[A_f^{lm}e_{3f}\zeta\right]\;\delta_j\left[1\right]\right\} \quad \equiv 0$$

C.7.2 Dissipation of Horizontal Kinetic Energy

The lateral momentum diffusion term dissipates the horizontal kinetic energy :

$$\int_D \mathbf{U}_h\cdot\left[\nabla_h\,\left(A^{lm}\,\chi\right)-\nabla_h\times\left(A^{lm}\,\zeta\,\mathbf{k}\right)\right]\,dv$$

$$\begin{aligned}
&\equiv \sum_{i,j,k}\left\{\frac{1}{e_{1u}}\delta_{i+1/2}\left[A_T^{lm}\chi\right]-\frac{1}{e_{2u}\,e_{3u}}\delta_j\left[A_f^{lm}e_{3f}\zeta\right]\right\}\;e_{1u}\,e_{2u}\,e_{3u}\,u\\
&\quad+\left\{\frac{1}{e_{2u}}\delta_{j+1/2}\left[A_T^{lm}\chi\right]+\frac{1}{e_{1v}\,e_{3v}}\delta_i\left[A_f^{lm}e_{3f}\zeta\right]\right\}\;e_{1v}\,e_{2u}\,e_{3v}\,v\\
&\equiv \sum_{i,j,k}\Big\{e_{2u}\,e_{3u}\,u\,\delta_{i+1/2}\left[A_T^{lm}\chi\right]-e_{1u}\,u\,\delta_j\left[A_f^{lm}e_{3f}\zeta\right]\Big\}\\
&\quad+\Big\{e_{1v}\,e_{3v}\,v\,\delta_{j+1/2}\left[A_T^{lm}\chi\right]+e_{2v}\,v\,\delta_i\left[A_f^{lm}e_{3f}\zeta\right]\Big\}\\
&\equiv \sum_{i,j,k}-\Big(\delta_i\left[e_{2u}\,e_{3u}\,u\right]+\delta_j\left[e_{1v}\,e_{3v}\,v\right]\Big)\,A_T^{lm}\chi\\
&\quad-\Big(\delta_{i+1/2}\left[e_{2v}\,v\right]-\delta_{j+1/2}\left[e_{1u}\,u\right]\Big)\,A_f^{lm}e_{3f}\zeta\\
&\equiv \sum_{i,j,k}-A_T^{lm}\,\chi^2\,e_{1t}\,e_{2t}\,e_{3t}-A_f^{lm}\,\zeta^2\,e_{1f}\,e_{2f}\,e_{3f} \quad \leq 0
\end{aligned}$$

C.7.3 Dissipation of Enstrophy

The lateral momentum diffusion term dissipates the enstrophy when the eddy coefficients are horizontally uniform :

$$
\begin{aligned}
&\int_D \zeta\, \mathbf{k} \cdot \nabla \times \left[\nabla_h \left(A^{lm}\, \chi \right) - \nabla_h \times \left(A^{lm}\, \zeta\, \mathbf{k} \right) \right] \, dv \\
&= A^{lm} \int_D \zeta \mathbf{k} \cdot \nabla \times \left[\nabla_h \times \left(\zeta\, \mathbf{k} \right) \right] \, dv \\
&\equiv A^{lm} \sum_{i,j,k} \zeta\; e_{3f} \left\{ \delta_{i+1/2} \left[\frac{e_{2v}}{e_{1v}\, e_{3v}} \delta_i \left[e_{3f} \zeta \right] \right] + \delta_{j+1/2} \left[\frac{e_{1u}}{e_{2u}\, e_{3u}} \delta_j \left[e_{3f} \zeta \right] \right] \right\}
\end{aligned}
$$

Using (4.11), it follows :

$$
\begin{aligned}
&\equiv -A^{lm} \sum_{i,j,k} \left\{ \left(\frac{1}{e_{1v}\, e_{3v}} \delta_i \left[e_{3f} \zeta \right] \right)^2 b_v + \left(\frac{1}{e_{2u}\, e_{3u}} \delta_j \left[e_{3f} \zeta \right] \right)^2 b_u \right\} \\
&\leq\; 0
\end{aligned}
$$

C.7.4 Conservation of Horizontal Divergence

When the horizontal divergence of the horizontal diffusion of momentum (discrete sense) is taken, the term associated with the vertical curl of the vorticity is zero locally, due to (! ! ! II.1.8 ! ! ! ! !). The resulting term conserves the χ and dissipates χ^2 when the eddy coefficients are horizontally uniform.

$$
\begin{aligned}
&\int_D \nabla_h \cdot \left[\nabla_h \left(A^{lm}\, \chi \right) - \nabla_h \times \left(A^{lm}\, \zeta\, \mathbf{k} \right) \right] dv = \int_D \nabla_h \cdot \nabla_h \left(A^{lm}\, \chi \right) dv \\
&\equiv \sum_{i,j,k} \left\{ \delta_i \left[A_u^{lm} \frac{e_{2u}\, e_{3u}}{e_{1u}} \delta_{i+1/2} \left[\chi \right] \right] + \delta_j \left[A_v^{lm} \frac{e_{1v}\, e_{3v}}{e_{2v}} \delta_{j+1/2} \left[\chi \right] \right] \right\}
\end{aligned}
$$

Using (4.11), it follows :

$$
\equiv \sum_{i,j,k} - \left\{ \frac{e_{2u}\, e_{3u}}{e_{1u}} A_u^{lm} \delta_{i+1/2} \left[\chi \right] \delta_{i+1/2} \left[1 \right] + \frac{e_{1v}\, e_{3v}}{e_{2v}} A_v^{lm} \delta_{j+1/2} \left[\chi \right] \delta_{j+1/2} \left[1 \right] \right\} \qquad \equiv 0
$$

C.7.5 Dissipation of Horizontal Divergence Variance

$$\int_D \chi \, \nabla_h \cdot \left[\nabla_h \left(A^{lm} \, \chi \right) - \nabla_h \times \left(A^{lm} \, \zeta \, \mathbf{k} \right) \right] \, dv = A^{lm} \int_D \chi \, \nabla_h \cdot \nabla_h \left(\chi \right) \, dv$$
$$\equiv A^{lm} \sum_{i,j,k} \frac{1}{e_{1t} \, e_{2t} \, e_{3t}} \chi \left\{ \delta_i \left[\frac{e_{2u} \, e_{3u}}{e_{1u}} \delta_{i+1/2} \left[\chi \right] \right] + \delta_j \left[\frac{e_{1v} \, e_{3v}}{e_{2v}} \delta_{j+1/2} \left[\chi \right] \right] \right\} \, e_{1t} \, e_{2t} \, e_{3t}$$

Using (4.11), it turns out to be :

$$\equiv -A^{lm} \sum_{i,j,k} \left\{ \left(\frac{1}{e_{1u}} \delta_{i+1/2} \left[\chi \right] \right)^2 b_u + \left(\frac{1}{e_{2v}} \delta_{j+1/2} \left[\chi \right] \right)^2 b_v \right\}$$
$$\leq 0$$

C.8 Conservation Properties on Vertical Momentum Physics

As for the lateral momentum physics, the continuous form of the vertical diffusion of momentum satisfies several integral constraints. The first two are associated with the conservation of momentum and the dissipation of horizontal kinetic energy :

$$\int_D \frac{1}{e_3} \, \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \, \frac{\partial \mathbf{U}_h}{\partial k} \right) \, dv \qquad = \vec{\mathbf{0}}$$

and

$$\int_D \mathbf{U}_h \cdot \frac{1}{e_3} \, \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \, \frac{\partial \mathbf{U}_h}{\partial k} \right) \, dv \quad \leq 0$$

The first property is obvious. The second results from :

$$\int_D \mathbf{U}_h \cdot \frac{1}{e_3} \, \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \, \frac{\partial \mathbf{U}_h}{\partial k} \right) \, dv$$

$$\equiv \sum_{i,j,k} \left(u \, \delta_k \left[\frac{A_u^{vm}}{e_{3uw}} \delta_{k+1/2} \left[u \right] \right] \, e_{1u} \, e_{2u} + v \, \delta_k \left[\frac{A_v^{vm}}{e_{3vw}} \delta_{k+1/2} \left[v \right] \right] \, e_{1v} \, e_{2v} \right)$$

since the horizontal scale factor does not depend on k, it follows :

$$\equiv -\sum_{i,j,k} \left(\frac{A_u^{vm}}{e_{3uw}} \left(\delta_{k+1/2} \left[u \right] \right)^2 \; e_{1u} \, e_{2u} + \frac{A_v^{vm}}{e_{3vw}} \left(\delta_{k+1/2} \left[v \right] \right)^2 \; e_{1v} \, e_{2v} \right) \quad \leq 0$$

The vorticity is also conserved. Indeed :

$$\int_D \frac{1}{e_3} \mathbf{k} \cdot \nabla \times \left(\frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \right) \, dv$$

$$\equiv \sum_{i,j,k} \frac{1}{e_{3f}} \frac{1}{e_{1f} \, e_{2f}} \Bigg\{ \quad \delta_{i+1/2} \left(\frac{e_{2v}}{e_{3v}} \delta_k \left[\frac{1}{e_{3vw}} \delta_{k+1/2} \left[v \right] \right] \right) - \delta_{j+1/2} \left(\frac{e_{1u}}{e_{3u}} \delta_k \left[\frac{1}{e_{3uw}} \delta_{k+1/2} \left[u \right] \right] \right) \Bigg\} \, e_{1f} \, e_{2f} \, e_{3f} \; \equiv 0$$

If the vertical diffusion coefficient is uniform over the whole domain, the enstrophy is dissipated, $i.e.$

$$\int_D \zeta \, \mathbf{k} \cdot \nabla \times \left(\frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \right) \, dv = 0$$

This property is only satisfied in z-coordinates :

$$\int_D \zeta \, \mathbf{k} \cdot \nabla \times \left(\frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \right) \, dv$$

$$\equiv \sum_{i,j,k} \zeta \; e_{3f} \left\{ \quad \delta_{i+1/2} \left(\frac{e_{2v}}{e_{3v}} \delta_k \left[\frac{A_v^{vm}}{e_{3vw}} \delta_{k+1/2} [v] \right] \right) - \delta_{j+1/2} \left(\frac{e_{1u}}{e_{3u}} \delta_k \left[\frac{A_u^{vm}}{e_{3uw}} \delta_{k+1/2} [u] \right] \right) \right\}$$

$$\equiv \sum_{i,j,k} \zeta \; e_{3f} \Bigg\{ \quad \frac{1}{e_{3v}} \delta_k \left[\frac{A_v^{vm}}{e_{3vw}} \delta_{k+1/2} \left[\delta_{i+1/2} \left[e_{2v} \, v \right] \right] \right] - \frac{1}{e_{3u}} \delta_k \left[\frac{A_u^{vm}}{e_{3uw}} \delta_{k+1/2} \left[\delta_{j+1/2} \left[e_{1u} \, u \right] \right] \right] \Bigg\}$$

Using the fact that the vertical diffusion coefficients are uniform, and that in z-coordinate, the vertical scale factors do not depend on i and j so that : $e_{3f} = e_{3u} = e_{3v} = e_{3t}$ and $e_{3w} = e_{3uw} = e_{3vw}$, it follows :

$$\equiv A^{vm} \sum_{i,j,k} \zeta \; \delta_k \left[\frac{1}{e_{3w}} \delta_{k+1/2} \Big[\delta_{i+1/2} \left[e_{2v} \, v \right] - \delta_{j+1/2} \left[e_{1u} \, u \right] \Big] \right]$$

$$\equiv -A^{vm} \sum_{i,j,k} \frac{1}{e_{3w}} \left(\delta_{k+1/2} \left[\zeta \right] \right)^2 \; e_{1f} \, e_{2f} \; \leq 0$$

Similarly, the horizontal divergence is obviously conserved :

$$\int_D \nabla \cdot \left(\frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \right) \; dv = 0$$

and the square of the horizontal divergence decreases ($i.e.$ the horizontal divergence is dissipated) if the vertical diffusion coefficient is uniform over the whole domain :

$$\int_D \chi \, \nabla \cdot \left(\frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \right) \; dv = 0$$

This property is only satisfied in the z-coordinate :

$$\int_D \chi \, \nabla \cdot \left(\frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \right) \; dv$$

$$\equiv \sum_{i,j,k} \frac{\chi}{e_{1t}\, e_{2t}} \Bigg\{ \quad \delta_{i+1/2} \left(\frac{e_{2u}}{e_{3u}} \delta_k \left[\frac{A_u^{vm}}{e_{3uw}} \delta_{k+1/2}[u] \right] \right) + \delta_{j+1/2} \left(\frac{e_{1v}}{e_{3v}} \delta_k \left[\frac{A_v^{vm}}{e_{3vw}} \delta_{k+1/2}[v] \right] \right) \Bigg\} \, e_{1t}\, e_{2t}\, e_{3t}$$

$$\equiv A^{vm} \sum_{i,j,k} \chi \Bigg\{ \quad \delta_{i+1/2} \left(\delta_k \left[\frac{1}{e_{3uw}} \delta_{k+1/2} \left[e_{2u}\, u \right] \right] \right) + \delta_{j+1/2} \left(\delta_k \left[\frac{1}{e_{3vw}} \delta_{k+1/2} \left[e_{1v}\, v \right] \right] \right) \Bigg\}$$

$$\equiv -A^{vm} \sum_{i,j,k} \frac{\delta_{k+1/2}\left[\chi\right]}{e_{3w}} \left\{ \delta_{k+1/2} \Big[\delta_{i+1/2} \left[e_{2u}\, u \right] + \delta_{j+1/2} \left[e_{1v}\, v \right] \Big] \right\}$$

$$\equiv -A^{vm} \sum_{i,j,k} \frac{1}{e_{3w}} \delta_{k+1/2} \left[\chi \right] \; \delta_{k+1/2} \left[e_{1t}\, e_{2t}\, \chi \right]$$

$$\equiv -A^{vm} \sum_{i,j,k} \frac{e_{1t}\, e_{2t}}{e_{3w}} \; \left(\delta_{k+1/2} \left[\chi \right] \right)^2 \quad \equiv 0$$

C.9 Conservation Properties on Tracer Physics

The numerical schemes used for tracer subgridscale physics are written such that the heat and salt contents are conserved (equations in flux form, second order centered finite differences). Since a flux form is used to compute the temperature and salinity, the quadratic form of these quantities (i.e. their variance) globally tends to diminish. As for the advection term, there is generally no strict conservation of mass, even if in practice the mass is conserved to a very high accuracy.

C.9.1 Conservation of Tracers

constraint of conservation of tracers :

$$\int_D \nabla \cdot (A\, \nabla T)\ dv$$

$$\equiv \sum_{i,j,k} \Bigg\{ \delta_i \left[A_u^{lT} \frac{e_{2u}\, e_{3u}}{e_{1u}} \delta_{i+1/2}\left[T\right] \right] + \delta_j \left[A_v^{lT} \frac{e_{1v}\, e_{3v}}{e_{2v}} \delta_{j+1/2}\left[T\right] \right] + \delta_k \left[A_w^{vT} \frac{e_{1t}\, e_{2t}}{e_{3t}} \delta_{k+1/2}\left[T\right] \right] \Bigg\} \quad \equiv 0$$

In fact, this property simply results from the flux form of the operator.

C.9.2 Dissipation of Tracer Variance

constraint on the dissipation of tracer variance :

$$\int_D T\ \nabla \cdot (A\, \nabla T)\ dv$$

$$\equiv \sum_{i,j,k} T \Bigg\{ \delta_i \left[A_u^{lT} \frac{e_{2u}\, e_{3u}}{e_{1u}} \delta_{i+1/2}\left[T\right] \right] + \delta_j \left[A_v^{lT} \frac{e_{1v}\, e_{3v}}{e_{2v}} \delta_{j+1/2}\left[T\right] \right] + \delta_k \left[A_w^{vT} \frac{e_{1t}\, e_{2t}}{e_{3t}} \delta_{k+1/2}\left[T\right] \right] \Bigg\}$$

$$\equiv -\sum_{i,j,k} \Bigg\{ A_u^{lT} \left(\frac{1}{e_{1u}} \delta_{i+1/2}\left[T\right] \right)^2 e_{1u}\, e_{2u}\, e_{3u} + A_v^{lT} \left(\frac{1}{e_{2v}} \delta_{j+1/2}\left[T\right] \right)^2 e_{1v}\, e_{2v}\, e_{3v} + A_w^{vT} \left(\frac{1}{e_{3w}} \delta_{k+1/2}\left[T\right] \right)^2 e_{1w}\, e_{2w}\, e_{3w} \Bigg\} \quad \leq 0$$

D Coding Rules

Contents

A ”model life” is more than ten years. Its software, composed of a few hundred modules, is used by many people who are scientists or students and do not necessarily know every aspect of computing very well. Moreover, a well thought-out program is easier to read and understand, less difficult to modify, produces fewer bugs and is easier to maintain. Therefore, it is essential that the model development follows some rules :

- well planned and designed
- well written
- well documented (both on- and off-line)
- maintainable
- easily portable
- flexible.

To satisfy part of these aims, *NEMO* is written with a coding standard which is close to the ECMWF rules, named DOCTOR [**?**]. These rules present some advantages like :

- to provide a well presented program
- to use rules for variable names which allow recognition of their type (integer, real, parameter, local or shared variables, etc.).

This facilitates both the understanding and the debugging of an algorithm.

D.1 The program structure

Each program begins with a set of headline comments containing :

- the program title
- the purpose of the routine
- the method and algorithms used
- the detail of input and output interfaces
- the external routines and functions used (if they exist)
- references (if they exist)
- the author name(s), the date of creation and any updates.
- Each program is split into several well separated sections and sub-sections with an underlined title and specific labelled statements.
- A program has not more than 200 to 300 lines.

A template of a module style can be found on the NEMO depository in the following file : NEMO/OPA_SRC/module_example.

D.2 Coding conventions

- Use of the universal language FORTRAN 90, and try to avoid obsolescent features like statement functions, do not use GO TO and EQUIVALENCE statements.

- A continuation line begins with the character & indented by three spaces compared to the previous line, while the previous line ended with the character &.

- All the variables must be declared. The code is usually compiled with implicit none.

- Never use continuation lines in the declaration of a variable. When searching a variable in the code through a *grep* command, the declaration line will be found.

- In the declaration of a PUBLIC variable, the comment part at the end of the line should start with the two characters "`!:`". the following UNIX command,

```
grep var_name *90 \ grep \!:
```

will display the module name and the line where the var_name declaration is.

- Always use a three spaces indentation in DO loop, CASE, or IF-ELSEIF-ELSE-ENDIF statements.

- use a space after a comma, except when it appears to separate the indices of an array.

- use call to ctl_stop routine instead of just a STOP.

D.3 Naming Conventions

The purpose of the naming conventions is to use prefix letters to classify model variables. These conventions allow the variable type to be easily known and rapidly identified. The naming conventions are summarised in the Table below :

Type / Status	integer	real	logical	character	structure	double precision	complex
public or module variable	**m n** *but not* **nn_**	**a b e f g h o q r t** *to* **x** but not **fs rn_**	**l** *but not* **lp ld ll ln_**	**c** *but not* **cp cd cl cn_**	**s** *but not* **sd sd sl sn_**	**d** *but not* **dp dd dl dn_**	**y** *but not* **yp yd yl yn**
dummy argument	**k** *but not* **kf**	**p** *but not* **pp pf**	**ld**	**cd**	**sd**	**dd**	**yd**
local variable	**i**	**z**	**ll**	**cl**	**sl**	**dl**	**yl**
loop control	**j** *but not* **jp**						
parameter	**jp**	**pp**	**lp**	**cp**	**sp**	**dp**	**yp**
namelist	**nn_**	**rn_**	**ln_**	**cn_**	**sn_**	**dn_**	**yn_**
CPP macro	**kf**	**fs**					

D.4 The program structure

To be done....

E Griffies's iso-neutral diffusion

Contents

E.1 Griffies's formulation of iso-neutral diffusion

E.1.1 Introduction

We define a scheme that get its inspiration from the scheme of **?**, but is formulated within the *NEMO* framework, using scale factors rather than grid-sizes.

The off-diagonal terms of the small angle diffusion tensor (2.37) produce skew-fluxes along the i- and j-directions resulting from the vertical tracer gradient :

$$+\kappa r_1 \frac{1}{e_3}\frac{\partial T}{\partial k}, \qquad +\kappa r_2 \frac{1}{e_3}\frac{\partial T}{\partial k} \tag{E.1}$$

and in the k-direction resulting from the lateral tracer gradients

$$\kappa r_1 \frac{1}{e_1}\frac{\partial T}{\partial i} + \kappa r_2 \frac{1}{e_1}\frac{\partial T}{\partial i} \tag{E.2}$$

where (2.38)

$$\begin{aligned} r_1 &= -\frac{e_3}{e_1}\left(\frac{\partial \rho}{\partial i}\right)\left(\frac{\partial \rho}{\partial k}\right)^{-1} \\ &= -\frac{e_3}{e_1}\left(-\alpha\frac{\partial T}{\partial i} + \beta\frac{\partial S}{\partial i}\right)\left(-\alpha\frac{\partial T}{\partial k} + \beta\frac{\partial S}{\partial k}\right)^{-1} \end{aligned}$$

is the i-component of the slope of the isoneutral surface relative to the computational surface, and r_2 is the j-component.

The extra vertical diffusive flux associated with the $_{33}$ component of the small angle diffusion tensor is

$$-\kappa(r_1^2 + r_2^2)\frac{1}{e_3}\frac{\partial T}{\partial k}. \tag{E.3}$$

Since there are no cross terms involving r_1 and r_2 in the above, we can consider the isoneutral diffusive fluxes separately in the i-k and j-k planes, just adding together the vertical components from each plane. The following description will describe the fluxes on the i-k plane.

There is no natural discretization for the i-component of the skew-flux, (E.1), as although it must be evaluated at u-points, it involves vertical gradients (both for the tracer and the slope r_1), defined at w-points. Similarly, the vertical skew flux, (E.2), is evaluated at w-points but involves horizontal gradients defined at u-points.

E.1.2 The standard discretization

The straightforward approach to discretize the lateral skew flux (E.1) from tracer cell i,k to $i+1,k$, introduced in 1995 into OPA, (5.10), is to calculate a mean vertical gradient at the u-point from the average of the four surrounding vertical tracer gradients, and multiply this by a mean slope at the u-point, calculated from the averaged surrounding vertical density gradients. The total area-integrated skew-flux from tracer cell i,k to $i+1,k$, noting that the $e_{3u^k_{i+1/2}}$ in the area $e_{3u^k_{i+1/2}} e_{2u^k_{i+1/2}}$ at the u-point cancels out with the $1/e_{3u^k_{i+1/2}}$ associated with the vertical tracer gradient, is then (5.10)

$$\left(F_u^{\text{skew}}\right)^k_{i+\frac{1}{2}} = \kappa^k_{i+\frac{1}{2}} e_{2u^k_{i+1/2}} \overline{r_1}^{\,i,k} \overline{\overline{\delta_k T}}^{\,i,k},$$

where

$$\overline{r_1}^{\,i,k} = -\frac{e_{3u^k_{i+1/2}}}{e_{1u^k_{i+1/2}}} \frac{\delta_{i+1/2}[\rho]}{\overline{\overline{\delta_k \rho}}^{\,i,k}}.$$

Unfortunately the resulting combination $\overline{\overline{\delta_k \bullet}}^{i,k}$ of a k average and a k difference reduces to $\bullet_{k+1} - \bullet_{k-1}$, so two-grid-point oscillations are invisible to this discretization of the iso-neutral operator. These *computational modes* will not be damped by this operator, and may even possibly be amplified by it. Consequently, applying this operator to a tracer does not guarantee the decrease of its global-average variance. To correct this, we introduced a smoothing of the slopes of the iso-neutral surfaces (see §9). This technique works fine for T and S as they are active tracers ($i.e.$ they enter the computation of density), but it does not work for a passive tracer.

E.1.3 Expression of the skew-flux in terms of triad slopes

[**?**] introduce a different discretization of the off-diagonal terms that nicely solves the problem. They get the skew flux from the products of the vertical gradients at each

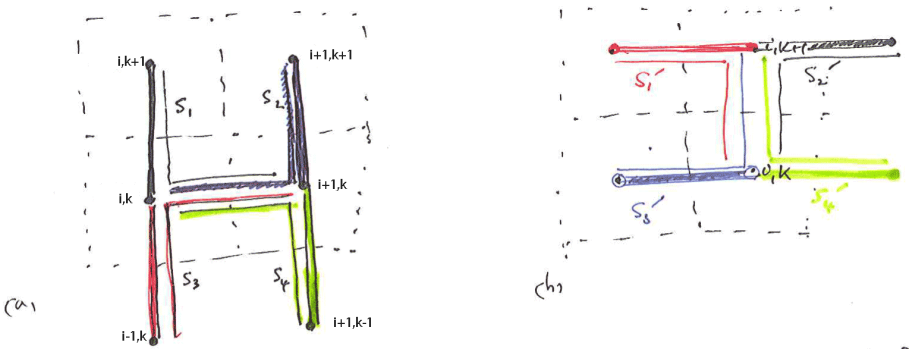

FIG. E.1 – (a) Arrangement of triads S_i and tracer gradients to give lateral tracer flux from box i, k to $i+1, k$ (b) Triads S_i' and tracer gradients to give vertical tracer flux from box i, k to $i, k+1$.

w-point surrounding the u-point with the corresponding 'triad' slope calculated from the lateral density gradient across the u-point divided by the vertical density gradient at the same w-point as the tracer gradient. See Fig. E.1a, where the thick lines denote the tracer gradients, and the thin lines the corresponding triads, with slopes $s_1, \ldots s_4$. The total area-integrated skew-flux from tracer cell i, k to $i+1, k$

$$\begin{aligned}\left(F_u^{13}\right)_{i+\frac{1}{2}}^{k} = {} & \kappa_{i+1}^{k} a_1 s_1 \delta_{k+\frac{1}{2}} \left[T^{i+1}\right] / e_{3w_{i+1}}^{k+\frac{1}{2}} + \kappa_i^k a_2 s_2 \delta_{k+\frac{1}{2}} \left[T^{i}\right] / e_{3w_{i+1}}^{k+\frac{1}{2}} \\ & + \kappa_{i+1}^{k} a_3 s_3 \delta_{k-\frac{1}{2}} \left[T^{i+1}\right] / e_{3w_{i+1}}^{k+\frac{1}{2}} + \kappa_i^k a_4 s_4 \delta_{k-\frac{1}{2}} \left[T^{i}\right] / e_{3w_{i+1}}^{k+\frac{1}{2}}, \end{aligned} \quad \text{(E.4)}$$

where the contributions of the triad fluxes are weighted by areas $a_1, \ldots a_4$, and κ is now defined at the tracer points rather than the u-points. This discretization gives a much closer stencil, and disallows the two-point computational modes.

The vertical skew flux (E.2) from tracer cell i,k to $i,k+1$ at the w-point $i,k+\frac{1}{2}$ is constructed similarly (Fig. E.1b) by multiplying lateral tracer gradients from each of the four surrounding u-points by the appropriate triad slope :

$$\left(F_w^{31}\right)_i^{k+\frac{1}{2}} = \kappa_i^{k+1} a_1' s_1' \delta_{i-\frac{1}{2}} \left[T^{k+1}\right] / e_{3u}{}_{i-\frac{1}{2}}^{k+1} + \kappa_i^{k+1} a_2' s_2' \delta_{i+\frac{1}{2}} \left[T^{k+1}\right] / e_{3u}{}_{i+\frac{1}{2}}^{k+1} + \kappa_i^{k} a_3' s_3' \delta_{i-\frac{1}{2}} \left[T^{k}\right] / e_{3u}{}_{i-\frac{1}{2}}^{k} + \kappa_i^{k} a_4' s_4' \delta_{i+\frac{1}{2}} \left[T^{k}\right] / e_{3u}{}_{i+\frac{1}{2}}^{k}. \quad \text{(E.5)}$$

We notate the triad slopes in terms of the 'anchor point' i,k (appearing in both the vertical and lateral gradient), and the u- and w-points $(i+i_p,k)$, $(i,k+k_p)$ at the centres of the 'arms' of the triad as follows (see also Fig. E.1) :

$${}_i^k\mathbb{R}_{i_p}^{k_p} = \frac{e_{3w}{}_i^{k+k_p}}{e_{1u}{}_{i+i_p}^{k}} \frac{(\alpha/\beta)_i^k \ \delta_{i+i_p}[T^k] - \delta_{i+i_p}[S^k]}{(\alpha/\beta)_i^k \ \delta_{k+k_p}[T^i] - \delta_{k+k_p}[S^i]}. \quad \text{(E.6)}$$

In calculating the slopes of the local neutral surfaces, the expansion coefficients α and β are evaluated at the anchor points of the triad [1], while the metrics are calculated at the u- and w-points on the arms.

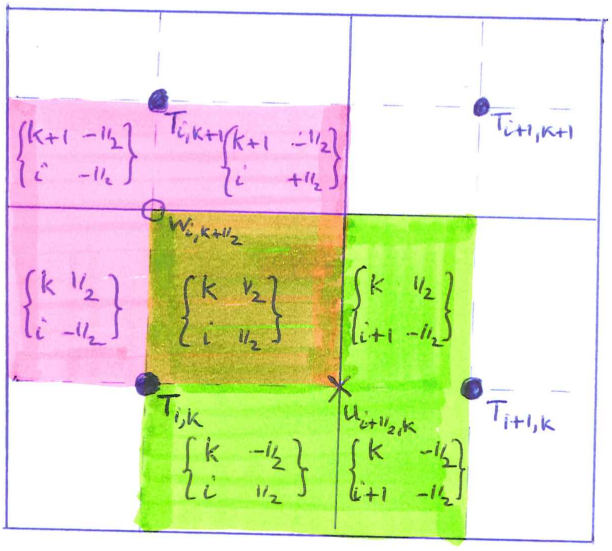

FIG. E.2 – Triad notation for quarter cells.T-cells are inside boxes, while the $i+\frac{1}{2},k$ u-cell is shaded in green and the $i,k+\frac{1}{2}$ w-cell is shaded in pink.

Each triad $\{{}_i^k \ {}_{i_p}^{k_p}\}$ is associated (Fig. E.2) with the quarter cell that is the intersection of the i,k T-cell, the $i+i_p,k$ u-cell and the $i,k+k_p$ w-cell. Expressing the slopes s_i and

[1] Note that in (E.6) we use the ratio α/β instead of multiplying the temperature derivative by α and the salinity derivative by β. This is more efficient as the ratio α/β can to be evaluated directly

s'_i in (E.4) and (E.5) in this notation, we have e.g. $s_1 = s'_1 = {}^{k}_{i}\mathbb{R}^{1/2}_{1/2}$. Each triad slope ${}^{k}_{i}\mathbb{R}^{k_p}_{i_p}$ is used once (as an s) to calculate the lateral flux along its u-arm, at $(i+i_p, k)$, and then again as an s' to calculate the vertical flux along its w-arm at $(i, k+k_p)$. Each vertical area a_i used to calculate the lateral flux and horizontal area a'_i used to calculate the vertical flux can also be identified as the area across the u- and w-arms of a unique triad, and we can notate these areas, similarly to the triad slopes, as ${}^{k}_{i}\mathbb{A}_{u}{}^{k_p}_{i_p}$, ${}^{k}_{i}\mathbb{A}_{w}{}^{k_p}_{i_p}$, where e.g. in (E.4) $a_1 = {}^{k}_{i}\mathbb{A}_{u}{}^{1/2}_{1/2}$, and in (E.5) $a'_1 = {}^{k}_{i}\mathbb{A}_{w}{}^{1/2}_{1/2}$.

E.1.4 The full triad fluxes

A key property of isoneutral diffusion is that it should not affect the (locally referenced) density. In particular there should be no lateral or vertical density flux. The lateral density flux disappears so long as the area-integrated lateral diffusive flux from tracer cell i, k to $i+1, k$ coming from the $_{11}$ term of the diffusion tensor takes the form

$$\left(F^{11}_{u}\right)^{k}_{i+\frac{1}{2}} = -\left(\kappa^{k+1}_{i} a_1 + \kappa^{k+1}_{i} a_2 + \kappa^{k}_{i} a_3 + \kappa^{k}_{i} a_4\right) \frac{\delta_{i+1/2}\left[T^k\right]}{e_{1u}{}^{k}_{i+1/2}}, \tag{E.7}$$

where the areas a_i are as in (E.4). In this case, separating the total lateral flux, the sum of (E.4) and (E.7), into triad components, a lateral tracer flux

$${}^{k}_{i}\mathbb{F}_{u}{}^{k_p}_{i_p}(T) = -A^{k}_{i}\, {}^{k}_{i}\mathbb{A}_{u}{}^{k_p}_{i_p} \left(\frac{\delta_{i+i_p}[T^k]}{e_{1u}{}^{k}_{i+i_p}} - {}^{k}_{i}\mathbb{R}^{k_p}_{i_p}\, \frac{\delta_{k+k_p}[T^i]}{e_{3w}{}^{k+k_p}_{i}} \right) \tag{E.8}$$

can be identified with each triad. Then, because the same metric factors $e_{3w}{}^{k+k_p}_{i}$ and $e_{1u}{}^{k}_{i+i_p}$ are employed for both the density gradients in ${}^{k}_{i}\mathbb{R}^{k_p}_{i_p}$ and the tracer gradients, the lateral density flux associated with each triad separately disappears.

$$\mathbb{F}_{u}{}^{k_p}_{i_p}(\rho) = -\alpha^{k}_{i}\, {}^{k}_{i}\mathbb{F}_{u}{}^{k_p}_{i_p}(T) + \beta^{k}_{i}\, {}^{k}_{i}\mathbb{F}_{u}{}^{k_p}_{i_p}(S) = 0 \tag{E.9}$$

Thus the total flux $\left(F^{31}_{u}\right)^{i}_{i,k+\frac{1}{2}} + \left(F^{11}_{u}\right)^{i}_{i,k+\frac{1}{2}}$ from tracer cell i, k to $i+1, k$ must also vanish since it is a sum of four such triad fluxes.

The squared slope r_1^2 in the expression (E.3) for the $_{33}$ component is also expressed in terms of area-weighted squared triad slopes, so the area-integrated vertical flux from tracer cell i, k to $i, k+1$ resulting from the r_1^2 term is

$$\left(F^{33}_{w}\right)^{k+\frac{1}{2}}_{i} = -\left(\kappa^{k+1}_{i} a'_1 s'^2_1 + \kappa^{k+1}_{i} a'_2 s'^2_2 + \kappa^{k}_{i} a'_3 s'^2_3 + \kappa^{k}_{i} a'_4 s'^2_4\right) \delta_{k+\frac{1}{2}}\left[T^{i+1}\right], \tag{E.10}$$

where the areas a' and slopes s' are the same as in (E.5). Then, separating the total vertical flux, the sum of (E.5) and (E.10), into triad components, a vertical flux

$${}^{k}_{i}\mathbb{F}_{w}{}^{k_p}_{i_p}(T) = A^{k}_{i}\, {}^{k}_{i}\mathbb{A}_{w}{}^{k_p}_{i_p} \left({}^{k}_{i}\mathbb{R}^{k_p}_{i_p}\, \frac{\delta_{i+i_p}[T^k]}{e_{1u}{}^{k}_{i+i_p}} - \left({}^{k}_{i}\mathbb{R}^{k_p}_{i_p}\right)^2 \frac{\delta_{k+k_p}[T^i]}{e_{3w}{}^{k+k_p}_{i}} \right) \tag{E.11}$$

$$= -\left({}^{k}_{i}\mathbb{A}_{w}{}^{k_p}_{i_p} \Big/ {}^{k}_{i}\mathbb{A}_{u}{}^{k_p}_{i_p} \right) {}^{k}_{i}\mathbb{R}^{k_p}_{i_p}\, {}^{k}_{i}\mathbb{F}_{u}{}^{k_p}_{i_p}(T) \tag{E.12}$$

may be associated with each triad. Each vertical density flux ${}^{k}_{i}\mathbb{F}_{w}{}^{k_p}_{i_p}(\rho)$ associated with a triad then separately disappears (because the lateral flux ${}^{k}_{i}\mathbb{F}_{u}{}^{k_p}_{i_p}(\rho)$ disappears). Consequently the total vertical density flux $\left(F^{31}_{w}\right)^{k+\frac{1}{2}}_{i}+\left(F^{33}_{w}\right)^{k+\frac{1}{2}}_{i}$ from tracer cell i,k to $i,k+1$ must also vanish since it is a sum of four such triad fluxes.

We can explicitly identify (Fig. E.2) the triads associated with the s_i, a_i, and s'_i, a'_i used in the definition of the u-fluxes and w-fluxes in (E.5), (E.4), (E.7) (E.10) and Fig. E.1 to write out the iso-neutral fluxes at u- and w-points as sums of the triad fluxes that cross the u- and w-faces :

$$\mathbf{F}_{iso}(T) \equiv \sum_{i_p,\,k_p} \begin{pmatrix} {}^{k}_{i+1/2-i_p}\mathbb{F}_{u}{}^{k_p}_{i_p}(T) \\ \\ {}^{k+1/2-k_p}_{i}\mathbb{F}_{w}{}^{k_p}_{i_p}(T) \end{pmatrix}. \tag{E.13}$$

E.1.5 Ensuring the scheme cannot increase tracer variance

We now require that this operator cannot increase the globally-integrated tracer variance. Each triad slope ${}^{k}_{i}\mathbb{R}^{k_p}_{i_p}$ drives a lateral flux ${}^{k}_{i}\mathbb{F}_{u}{}^{k_p}_{i_p}(T)$ across the u-point $i+i_p,k$ and a vertical flux ${}^{k}_{i}\mathbb{F}_{w}{}^{k_p}_{i_p}(T)$ across the w-point $i,k+k_p$. The lateral flux drives a net rate of change of variance at points $i+i_p-\frac{1}{2},k$ and $i+i_p+\frac{1}{2},k$ of

$$\begin{aligned} b_{T}{}^{k}_{i+i_p-1/2}\left(\frac{\partial T}{\partial t}T\right)^{k}_{i+i_p-1/2} \;+\; & b_{T}{}^{k}_{i+i_p+1/2}\left(\frac{\partial T}{\partial t}T\right)^{k}_{i+i_p+1/2} \\ &= -T^{k}_{i+i_p-1/2}\;{}^{k}_{i}\mathbb{F}_{u}{}^{k_p}_{i_p}(T) \;+\; T^{k}_{i+i_p+1/2}\;{}^{k}_{i}\mathbb{F}_{u}{}^{k_p}_{i_p}(T) \\ &= {}^{k}_{i}\mathbb{F}_{u}{}^{k_p}_{i_p}(T)\,\delta_{i+i_p}[T^k], \end{aligned} \tag{E.14}$$

while the vertical flux similarly drives a net rate of change of variance at points $i,k+k_p-\frac{1}{2}$ and $i,k+k_p+\frac{1}{2}$ of

$${}^{k}_{i}\mathbb{F}_{w}{}^{k_p}_{i_p}(T)\,\delta_{k+k_p}[T^i]. \tag{E.15}$$

The total variance tendency driven by the triad is the sum of these two. Expanding ${}^{k}_{i}\mathbb{F}_{u}{}^{k_p}_{i_p}(T)$ and ${}^{k}_{i}\mathbb{F}_{w}{}^{k_p}_{i_p}(T)$ with (E.8) and (E.11), it is

$$\begin{aligned} -A^{k}_{i}\Bigg\{ {}^{k}_{i}\mathbb{A}_{u}{}^{k_p}_{i_p}\left(\frac{\delta_{i+i_p}[T^k]}{e_{1u}{}^{k}_{i+i_p}}-{}^{k}_{i}\mathbb{R}^{k_p}_{i_p}\,\frac{\delta_{k+k_p}[T^i]}{e_{3w}{}^{k+k_p}_{i}}\right)\delta_{i+i_p}[T^k] \\ -{}^{k}_{i}\mathbb{A}_{w}{}^{k_p}_{i_p}\left(\frac{\delta_{i+i_p}[T^k]}{e_{1u}{}^{k}_{i+i_p}}-{}^{k}_{i}\mathbb{R}^{k_p}_{i_p}\,\frac{\delta_{k+k_p}[T^i]}{e_{3w}{}^{k+k_p}_{i}}\right){}^{k}_{i}\mathbb{R}^{k_p}_{i_p}\delta_{k+k_p}[T^i]\Bigg\}. \end{aligned}$$

The key point is then that if we require ${}^{k}_{i}\mathbb{A}_{u}{}^{k_p}_{i_p}$ and ${}^{k}_{i}\mathbb{A}_{w}{}^{k_p}_{i_p}$ to be related to a triad volume ${}^{k}_{i}\mathbb{V}^{k_p}_{i_p}$ by

$${}^{k}_{i}\mathbb{V}^{k_p}_{i_p} = {}^{k}_{i}\mathbb{A}_{u}{}^{k_p}_{i_p}\,e_{1u}{}^{k}_{i+i_p} = {}^{k}_{i}\mathbb{A}_{w}{}^{k_p}_{i_p}\,e_{3w}{}^{k+k_p}_{i}, \tag{E.16}$$

the variance tendency reduces to the perfect square

$$-A_i^k \, {}_i^k\mathbb{V}_{i_p}^{k_p} \left(\frac{\delta_{i+i_p}[T^k]}{e_{1u}{}_{i+i_p}^{k}} - {}_i^k\mathbb{R}_{i_p}^{k_p} \frac{\delta_{k+k_p}[T^i]}{e_{3w}{}_i^{k+k_p}} \right)^2 \leq 0. \tag{E.17}$$

Thus, the constraint (E.16) ensures that the fluxes associated with a given slope triad ${}_i^k\mathbb{R}_{i_p}^{k_p}$ do not increase the net variance. Since the total fluxes are sums of such fluxes from the various triads, this constraint, applied to all triads, is sufficient to ensure that the globally integrated variance does not increase.

The expression (E.16) can be interpreted as a discretization of the global integral

$$\frac{\partial}{\partial t} \int \tfrac{1}{2} T^2 \, dV = \int \mathbf{F} \cdot \nabla T \, dV, \tag{E.18}$$

where, within each triad volume ${}_i^k\mathbb{V}_{i_p}^{k_p}$, the lateral and vertical fluxes/unit area

$$\mathbf{F} = \left({}_i^k\mathbb{F}_{u\,i_p}^{k_p}(T)/{}_i^k\mathbb{A}_{u\,i_p}^{k_p},\ {}_i^k\mathbb{F}_{w\,i_p}^{k_p}(T)/{}_i^k\mathbb{A}_{w\,i_p}^{k_p} \right)$$

and the gradient

$$\nabla T = \left(\delta_{i+i_p}[T^k]/e_{1u}{}_{i+i_p}^{k},\, \delta_{k+k_p}[T^i]/e_{3w}{}_i^{k+k_p} \right)$$

E.1.6 Triad volumes in Griffes's scheme and in *NEMO*

To complete the discretization we now need only specify the triad volumes ${}_i^k\mathbb{V}_{i_p}^{k_p}$. **?** identify these ${}_i^k\mathbb{V}_{i_p}^{k_p}$ as the volumes of the quarter cells, defined in terms of the distances between T, u,f and w-points. This is the natural discretization of (E.18). The *NEMO* model, however, operates with scale factors instead of grid sizes, and scale factors for the quarter cells are not defined. Instead, therefore we simply choose

$${}_i^k\mathbb{V}_{i_p}^{k_p} = \tfrac{1}{4} b_{u}{}_{i+i_p}^{k}, \tag{E.19}$$

as a quarter of the volume of the u-cell inside which the triad quarter-cell lies. This has the nice property that when the slopes $\mathbb{R}$ vanish, the lateral flux from tracer cell i,k to $i+1,k$ reduces to the classical form

$$-\overline{A}^k_{i+1/2} \, \frac{b_{u}{}_{i+1/2}^{k}}{e_{1u}{}_{i+i_p}^{k}} \, \frac{\delta_{i+1/2}[T^k]}{e_{1u}{}_{i+i_p}^{k}} = -\overline{A}^k_{i+1/2} \, \frac{e_{1w}{}_{i+1/2}^{k} \, e_{1v}{}_{i+1/2}^{k}}{e_{1u}{}_{i+1/2}^{k}}. \tag{E.20}$$

In fact if the diffusive coefficient is defined at u-points, so that we employ $A^k_{i+i_p}$ instead of A^k_i in the definitions of the triad fluxes (E.8) and (E.11), we can replace $\overline{A}^k_{i+1/2}$ by $A^k_{i+1/2}$ in the above.

E.1.7 Summary of the scheme

The divergence of the expression (E.13) for the fluxes gives the iso-neutral diffusion tendency at each tracer point :

$$D_l^T = \frac{1}{b_T} \sum_{i_p,\, k_p} \left\{ \delta_i \left[{}^{k}_{i+1/2-i_p}\mathbb{F}_u{}^{k_p}_{i_p} \right] + \delta_k \left[{}^{k+1/2-k_p}_{i}\mathbb{F}_w{}^{k_p}_{i_p} \right] \right\} \tag{E.21}$$

where $b_T = e_{1T}\, e_{2T}\, e_{3T}$ is the volume of T-cells. The diffusion scheme satisfies the following six properties :

- **horizontal diffusion** The discretization of the diffusion operator recovers (E.20) the traditional five-point Laplacian in the limit of flat iso-neutral direction :

$$D_l^T = \frac{1}{b_T}\, \delta_i \left[\frac{e_{2u}\, e_{3u}}{e_{1u}}\, \overline{A}^{\,i}\, \delta_{i+1/2}[T] \right] \qquad \text{when} \quad {}^{k}_{i}\mathbb{R}^{k_p}_{i_p} = 0 \tag{E.22}$$

- **implicit treatment in the vertical** Only tracer values associated with a single water column appear in the expression (E.10) for the $_{33}$ fluxes, vertical fluxes driven by vertical gradients. This is of paramount importance since it means that an implicit in time algorithm can be used to solve the vertical diffusion equation. This is a necessity since the vertical eddy diffusivity associated with this term,

$$\frac{1}{b_w} \sum_{i_p,\, k_p} \left\{ {}^{k}_{i}\mathbb{V}^{k_p}_{i_p}\, A_i^k \left({}^{k}_{i}\mathbb{R}^{k_p}_{i_p} \right)^2 \right\} = \frac{1}{4b_w} \sum_{i_p,\, k_p} \left\{ b_u{}^{k}_{i+i_p}\, A_i^k \left({}^{k}_{i}\mathbb{R}^{k_p}_{i_p} \right)^2 \right\}, \tag{E.23}$$

  (where $b_w = e_{1w}\, e_{2w}\, e_{3w}$ is the volume of w-cells) can be quite large.

- **pure iso-neutral operator** The iso-neutral flux of locally referenced potential density is zero. See (E.9) and (E.12).

- **conservation of tracer** The iso-neutral diffusion conserves tracer content, $i.e.$

$$\sum_{i,j,k} \left\{ D_l^T\, b_T \right\} = 0 \tag{E.24}$$

  This property is trivially satisfied since the iso-neutral diffusive operator is written in flux form.

- **no increase of tracer variance** The iso-neutral diffusion does not increase the tracer variance, $i.e.$

$$\sum_{i,j,k} \left\{ T\, D_l^T\, b_T \right\} \leq 0 \tag{E.25}$$

  The property is demonstrated in E.1.5 above. It is a key property for a diffusion term. It means that it is also a dissipation term, $i.e.$ it is a diffusion of the square of the quantity on which it is applied. It therefore ensures that, when the diffusivity coefficient is large enough, the field on which it is applied become free of grid-point noise.

• **self-adjoint operator** The iso-neutral diffusion operator is self-adjoint, $i.e.$

$$\sum_{i,j,k} \{S\ D_l^T\ b_T\} = \sum_{i,j,k} \{D_l^S\ T\ b_T\} \tag{E.26}$$

In other word, there is no need to develop a specific routine from the adjoint of this operator. We just have to apply the same routine. This property can be demonstrated similarly to the proof of the 'no increase of tracer variance' property. The contribution by a single triad towards the left hand side of (E.26), can be found by replacing $\delta[T]$ by $\delta[S]$ in (E.14) and (E.14). This results in a term similar to (E.17),

$$-A_i^k\ {}_i^k\mathbb{V}_{i_p}^{k_p} \left(\frac{\delta_{i+i_p}[T^k]}{e_{1u}\ {}_{i+i_p}^{k}} - {}_i^k\mathbb{R}_{i_p}^{k_p} \frac{\delta_{k+k_p}[T^i]}{e_{3w}\ {}_{i}^{k+k_p}} \right) \left(\frac{\delta_{i+i_p}[S^k]}{e_{1u}\ {}_{i+i_p}^{k}} - {}_i^k\mathbb{R}_{i_p}^{k_p} \frac{\delta_{k+k_p}[S^i]}{e_{3w}\ {}_{i}^{k+k_p}} \right). \tag{E.27}$$

This is symmetrical in T and S, so exactly the same term arises from the discretization of this triad's contribution towards the RHS of (E.26).

E.2 Eddy induced velocity and Skew flux formulation

When Gent and McWilliams's [1990] diffusion is used (**key_traldf_eiv** defined), an additional advection term is added. The associated velocity is the so called eddy induced velocity, the formulation of which depends on the slopes of iso- neutral surfaces. Contrary to the case of iso-neutral mixing, the slopes used here are referenced to the geopotential surfaces, $i.e.$ (9.2) is used in z-coordinate, and the sum (9.2) + (9.3) in z^* or s-coordinates.

The eddy induced velocity is given by :

$$\begin{aligned} u^* &= -\frac{1}{e_2 e_3}\ \partial_k \left(e_2\ A_e\ r_i \right) \\ v^* &= -\frac{1}{e_1 e_3}\ \partial_k \left(e_1\ A_e\ r_j \right) \\ w^* &= \frac{1}{e_1 e_2}\ \left\{ \partial_i \left(e_2\ A_e\ r_i \right) + \partial_j \left(e_1\ A_e\ r_j \right) \right\} \end{aligned} \tag{E.28}$$

where A_e is the eddy induced velocity coefficient, and r_i and r_j the slopes between the iso-neutral and the geopotential surfaces.

The traditional way to implement this additional advection is to add it to the Eulerian velocity prior to computing the tracer advection. This allows us to take advantage of all the advection schemes offered for the tracers (see §5.1) and not just a 2^{nd} order advection scheme. This is particularly useful for passive tracers where *positivity* of the advection scheme is of paramount importance.

? introduces another way to implement the eddy induced advection, the so-called skew form. It is based on a transformation of the advective fluxes using the non-divergent

nature of the eddy induced velocity. For example in the (**i**,**k**) plane, the tracer advective fluxes can be transformed as follows :

$$\mathbf{F}^T_{eiv} = \begin{pmatrix} e_2\, e_3\, u^* \\ e_1\, e_2\, w^* \end{pmatrix} T = \begin{pmatrix} -\partial_k \left(e_2\, A_e\, r_i \right) T \\ +\partial_i \left(e_2\, A_e\, r_i \right) T \end{pmatrix} = \begin{pmatrix} -\partial_k \left(e_2\, A_e\, r_i\, T \right) \\ +\partial_i \left(e_2\, A_e\, r_i\, T \right) \end{pmatrix} + \begin{pmatrix} +e_2\, A_e\, r_i\, \partial_k T \\ -e_2\, A_e\, r_i\, \partial_i T \end{pmatrix}$$

and since the eddy induces velocity field is no-divergent, we end up with the skew form of the eddy induced advective fluxes :

$$\mathbf{F}^T_{eiv} = \begin{pmatrix} +e_2\, A_e\, r_i\, \partial_k T \\ -e_2\, A_e\, r_i\, \partial_i T \end{pmatrix} \tag{E.29}$$

The tendency associated with eddy induced velocity is then simply the divergence of the (E.29) fluxes. It naturally conserves the tracer content, as it is expressed in flux form. It also preserve the tracer variance.

The discrete form of (E.29) using the slopes (E.6) and defining A_e at T-point is then given by :

$$\mathbf{F}^T_{eiv} \equiv \sum_{i_p,\, k_p} \begin{pmatrix} +e_{2u}{}^{k}_{i+1/2-i_p}\; A_e{}^{k}_{i+1/2-i_p}\; {}^{k}_{i+1/2-i_p}\mathbb{R}^{k_p}_{i_p}\; \delta_{k+k_p}[T_{i+1/2-i_p}] \\ \\ -e_{2u}{}^{k+1/2-k_p}_{i}\; A_e{}^{k+1/2-k_p}_{i}\; {}^{k+1/2-k_p}_{i}\mathbb{R}^{k_p}_{i_p}\; \delta_{i+i_p}[T^{k+1/2-k_p}] \end{pmatrix} \tag{E.30}$$

Note that (E.30) is valid in z-coordinate with or without partial cells. In z^* or s-coordinate, the the slope between the level and the geopotential surfaces must be added to $\mathbb{R}$ for the discret form to be exact.

Such a choice of discretisation is consistent with the iso-neutral operator as it uses the same definition for the slopes. It also ensures the conservation of the tracer variance (see Appendix E.2.1), *i.e.* it does not include a diffusive component but is a 'pure' advection term.

E.2.1 Discrete Invariants of the skew flux formulation

Demonstration for the conservation of the tracer variance in the (**i**,**j**) plane.
This have to be moved in an Appendix.
The continuous property to be demonstrated is :

$$\int_D \nabla \cdot \mathbf{F}_{eiv}(T)\, T\, dv \equiv 0$$

The discrete form of its left hand side is obtained using (E.30)

$$\sum_{i,k} \sum_{i_p,\, k_p} \Bigg\{ \delta_i \Big[e_{2u}{}^{k}_{i+i_p+1/2} \quad A_{e}{}^{k}_{i+i_p+1/2} \quad {}^{k}_{i+i_p+1/2}\mathbb{R}^{k_p}_{-i_p} \quad \delta_{k+k_p}[T_{i+i_p+1/2}] \Big] \; T^k_i$$
$$- \delta_k \Big[e_{2u}{}^{k+k_p+1/2}_{i} \quad A_{e}{}^{k+k_p+1/2}_{i} \quad {}^{k+k_p+1/2}_{i}\mathbb{R}^{-k_p}_{i_p} \quad \delta_{i+i_p}[T^{k+k_p+1/2}] \Big] \; T^k_i \Bigg\}$$

apply the adjoint of delta operator, it becomes

$$\sum_{i,k} \sum_{i_p,\, k_p} \Bigg\{ \Big(e_{2u}{}^{k}_{i+i_p+1/2} \quad A_{e}{}^{k}_{i+i_p+1/2} \quad {}^{k}_{i+i_p+1/2}\mathbb{R}^{k_p}_{-i_p} \quad \delta_{k+k_p}[T_{i+i_p+1/2}] \Big) \; \delta_{i+1/2}[T^k]$$
$$- \Big(e_{2u}{}^{k+k_p+1/2}_{i} \quad A_{e}{}^{k+k_p+1/2}_{i} \quad {}^{k+k_p+1/2}_{i}\mathbb{R}^{-k_p}_{i_p} \quad \delta_{i+i_p}[T^{k+k_p+1/2}] \Big) \; \delta_{k+1/2}[T_i] \Bigg\}$$

Expending the summation on i_p and k_p, it becomes :

$$\sum_{i,k} \Big\{ \quad +e_{2u}{}^{k}_{i+1} \quad A_{e}{}^{k}_{i+1} \quad {}^{k}_{i+1}\mathbb{R}^{-1/2}_{-1/2} \quad \delta_{k-1/2}[T_{i+1}] \quad \delta_{i+1/2}[T^k]$$
$$+e_{2u}{}^{k}_{i} \quad A_{e}{}^{k}_{i} \quad {}^{k}_{i}\mathbb{R}^{-1/2}_{+1/2} \quad \delta_{k-1/2}[T_{i} \quad] \quad \delta_{i+1/2}[T^k]$$
$$+e_{2u}{}^{k}_{i+1} \quad A_{e}{}^{k}_{i+1} \quad {}^{k}_{i+1}\mathbb{R}^{+1/2}_{-1/2} \quad \delta_{k+1/2}[T_{i+1}] \quad \delta_{i+1/2}[T^k]$$
$$+e_{2u}{}^{k}_{i} \quad A_{e}{}^{k}_{i} \quad {}^{k}_{i}\mathbb{R}^{+1/2}_{+1/2} \quad \delta_{k+1/2}[T_{i} \quad] \quad \delta_{i+1/2}[T^k]$$
$$-e_{2u}{}^{k+1}_{i} \quad A_{e}{}^{k+1}_{i} \quad {}^{k+1}_{i}\mathbb{R}^{-1/2}_{-1/2} \quad \delta_{i-1/2}[T^{k+1}] \quad \delta_{k+1/2}[T_i]$$
$$-e_{2u}{}^{k}_{i} \quad A_{e}{}^{k}_{i} \quad {}^{k}_{i}\mathbb{R}^{+1/2}_{-1/2} \quad \delta_{i-1/2}[T^{k} \quad] \quad \delta_{k+1/2}[T_i]$$
$$-e_{2u}{}^{k+1}_{i} \quad A_{e}{}^{k+1}_{i} \quad {}^{k+1}_{i}\mathbb{R}^{-1/2}_{+1/2} \quad \delta_{i+1/2}[T^{k+1}] \quad \delta_{k+1/2}[T_i]$$
$$-e_{2u}{}^{k}_{i} \quad A_{e}{}^{k}_{i} \quad {}^{k}_{i}\mathbb{R}^{+1/2}_{+1/2} \quad \delta_{i+1/2}[T^{k} \quad] \quad \delta_{k+1/2}[T_i] \quad \Big\}$$

The two terms associated with the triad ${}^{k}_{i}\mathbb{R}^{+1/2}_{+1/2}$ are the same but of opposite signs, they cancel out. Exactly the same thing occurs for the triad ${}^{k}_{i}\mathbb{R}^{-1/2}_{-1/2}$. The two terms associated with the triad ${}^{k}_{i}\mathbb{R}^{-1/2}_{+1/2}$ are the same but both of opposite signs and shifted by 1 in k direction. When summing over k they cancel out with the neighbouring grid points. Exactly the same thing occurs for the triad ${}^{k}_{i}\mathbb{R}^{+1/2}_{-1/2}$ in the i direction. Therefore the sum over the domain is zero, $i.e.$ the variance of the tracer is preserved by the discretisation of the skew fluxes.